An introduction to Siconos

Vincent Acary, Olivier Bonnefon, Maurice Brémond, Olivier Huber, Franck Pérignon, Stephen Sinclair

An introduction to Siconos

Vincent Acary, Olivier Bonnefon, Maurice Brémond, Olivier Huber, Franck Pérignon, Stephen Sinclair

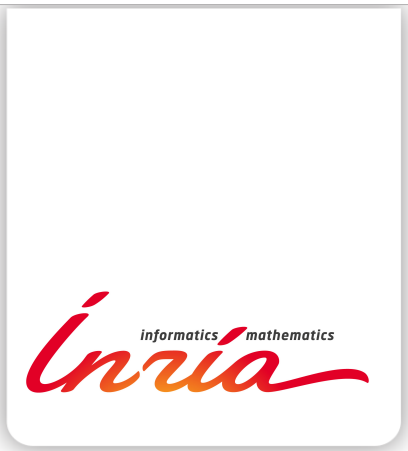
RESEARCH REPORT

N° 0340

July 2007

Project-Team Tripop

ISSN 0249-6399 ISRN INRIA/RR--0340--FR+ENG



An introduction to Siconos

Vincent Acary*, Olivier Bonnefon †, Maurice Brémond*, Olivier Huber‡, Franck Pérignon*, Stephen Sinclair§

Project-Team Tripop

Research Report n° 0340 — version 2 — initial version July 2007 — revised version January 2018 — 93 pages

Abstract: In this document, a brief overview of the Siconos Platform is given. One of the goal is to give a flavor on a simple example of the ability of the platform to model and simulate the so-called non smooth dynamical systems (NSDS). In particular, some examples of Lagrangian mechanical systems with contact and friction or electrical circuits with ideal and piecewise linear or ideal components (diodes, MOS transistors, ...) are commented. Then, the Siconos software is presented, starting from its architecture to a non exhaustive presentation of its components and functionalities. The aim of this document is not to serve as a reference guide but more as a illustrative introduction document to promote the use of the platform.

Key-words: Nonsmooth dynamical systems, linear complementarity systems, Mechanical Systems, unilateral contact, Coulomb friction, electrical circuits, ideal and piecewise linear components, siconos, event-capturing time–stepping schemes, event-detecting time–stepping schemes, nonsmooth law, set-valued law.

* Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP*, LJK, 38000 Grenoble, France
† INRA, Unité de Biostatistiques et Processus Spatiaux, Avignon France.
‡ Humboldt-Universität zu Berlin, Institut füür Mathematik, Berlin, Germany.
§ Universidad Rey Juan Carlos, Madrid, Spain.

RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Une introduction à Siconos

Résumé : Dans ce document, une brève introduction au logiciel Siconos est donnée. L'objectif est d'illustrer sur des exemples simples les capacités du logiciel à modéliser et simuler les systèmes dynamiques non lisses. En particulier, des exemples de systèmes lagrangiens avec contact et frottement et des circuits avec des éléments linéaires par morceaux ou idéaux (diodes transistors, ...) sont développés. Ensuite, le logiciel Siconos est présenté en détail, en commençant par son architecture jusqu'à une présentation détaillée d'une partie de ses composants et fonctionnalités. L'objectif de ce document n'est pas d'être une guide de référence du logiciel mais une introduction illustrant et promouvant l'utilisation de Siconos.

Mots-clés : Systèmes dynamiques non lisses, systèmes de complémentarité linéaire, systèmes mécaniques, contact unilatéral, forttement de Coulomb, circuits électriques, composants électriques idéaux ou linéaire par morceaux, Siconos, schémas à capture d'événements, schémas à détection d'événements, loi non lisse, loi multi-valuée.

Contents

Introduction

Siconos aims at providing a general and common tool for nonsmooth problems in various scientific fields like Applied Mathematics, Mechanics, Control Theory, Electrical circuits, Robotics, ... Most of the algorithm (mostly solvers for nonsmooth optimization problems) are written in C, whereas the modeling and simulation part is in C++. Fortran programs with compatible licenses are used for some integration routine. One of the design principle is to reuse existing and reliable software as building block. For instance, strong collaborations exist with HuMAns (humanoid motion modeling and control[1]) or LMGC90 (multi-body contact mechanics[2]) software package. We also have Python bindings, enabling the use of the Siconos platform inside the dynamic Python ecosystem.

For the end-user, there are 3 possible ways to build a simulation: write a C++ program, write a Python script or provide an xml file. In this document, for the sake of brevity, ease of use and clarity, we shall only provide Python code snippet. Through this document, a simple, basic but insightful example is used to illustrate the modeling and simulation process: the bouncing ball. It will also be used to highlight the differences between the various integration methods. In Part I, we simulate it in its simplest form. In Part II, we present the concept of NonSmooth Dynamical System (NSDS) and give the possible strategies to simulate it. In Part III, the general modeling and simulation principles in Siconos are explained. This part ends with a description of the main basic components, which help the user to build a NSDS in Siconos. Finally, in Part V, illustrative examples are presented in Mechanics, Control, Biology, Electrical Circuits.

Notations For $x, y \in \mathbb{R}^m$, the relation $x \perp y$ is equivalent to $\langle x, y \rangle = 0$ with $\langle \cdot , \cdot \rangle$ the usual inner product.
Font like `NonSmoothDynamicalSystem` will be used for Siconos objects (either C++ or Python). A complete documentation for each of these object can be found in Siconos Doxygen auto-generated documentation `http://siconos.gforge.inria.fr/Kernel`.

[1] `http://bipop.inrialpes.fr/software/humans/index.html`
[2] `http://www.lmgc.univ-montp2.fr/~dubois/LMGC90/`

Notes/Remarks Maurice/Franck Points to discuss: - examples : everything in Python and explain that C++ is available (links to Siconos Examples page?) - remove BouncingBall and DiodeBridge from last part : already done in first part. - other examples : no implementation details. Only a short presentation of the use case, some simulation results and maybe a python code for the key points (like a specific formulation for the OSNS ...) - remove section about Siconos components (Kernel, Numerics ...) from part II? Maybe we can write another part much more devel-like with details about Siconos Design and specific topics, like ideas about graphs in Siconos, smart pointers, W matrix assembly ... - leave or remove xml-like examples?

Part I

An insight into Siconos

Siconos is a library which provides objects and functions to model and simulate a nonsmooth problem. Therefore, solving such a problem with Siconos mainly consists in defining a set of objects in a Python or C++ "driver" file. The present part is dedicated to a short presentation of the general writing process of this driver, through two simple examples : a Lagrangian mechanical system, the bouncing ball, and an electrical, first order, system, the diode bridge. The point is to present the main steps required to model and simulate a system and to introduce the concepts in connection with "nonsmooth dynamical systems". Details will be provided in parts II and III.

For simplicity's sake, all examples are written in Python but most of them are available in their C++ form on Siconos Examples page at `http://siconos.gforge.inria.fr/Examples`.

1 The bouncing ball

The considered exeample is a ball of mass m and radius R, described by its generalized coordinates $q = (z)$, subjected to the gravity g and moving above a rigid plane, defined by its position h with respect to the axis Oz. The position of the plane is assumed to be fixed. See notations on Figure 1

Figure 1: A ball bouncing on the ground.

The first mandatory step is the definition of a `Model`, gathering a nonsmooth problem (the `NonSmoothDynamicalSystem` and the strategy to solve it (the `Simulation`).

```
import Siconos.Kernel as SK
# The model sets the time range for the simulation
tstart = 0. ; tend = 10.
bouncingBall = SK.Model(tstart, tend)
```

Note the first import : in Python, Siconos Kernel is available through the package *Siconos.Kernel*[3].

[3]See details about Siconos components, install process and so on in Part IV

Vector or matrix-like arguments can be defined as numpy arrays or as lists. Both cases appear in the examples below. As usual in Python, interactive help is available for each component.

```
help(SK.Model)

class Model(__builtin__.object)
|  Model: object that links the NonSmoothDynamicalSystem with a Simulation.
|
...
```

Then, the dynamics of the considered systems must be written. The equations of motion of a rigid ball bouncing on the ground are written as a linear Lagrangian time invariant dynamical system :

$$M\ddot{q}(t) = F_{ext} + p$$
$$v(t) = \dot{q}(t)$$
$$q(t_0) = q_0, v(t_0) = v_0$$

with M the inertia term, p the force due to the non-smooth law, i.e., the reaction at the impact times and $F_{ext}(t) : \mathcal{R} \mapsto \mathcal{R}^n$ the given external force.

Such a dynamical system corresponds to the `LagrangianLinearTIDS` (TIDS stands for Time-Invariant Dynamical System) class in Siconos.

```
radius = 0.1 # Radius of the ball
position = [1]  # initial position vector
velocity = [0]  # initial velocity vector
mass = 1.
# mass matrix
M = [[mass]]
ball = SK.LagrangianLinearTIDS(position, velocity, mass)
# Gravity must be applied through external forces:
g = 9.81
weight = [- mass * g]
ball.setFExtPtr(weight)
# The dynamical system must be inserted into the NSDS of the model:
bouncingBall.nonSmoothDynamicalSystem().insertDynamicalSystem(ball)
```

Dynamical systems may be subjected to some constraints (think of a set of rigid bodies that may interact), leading to a nonsmooth behavior. In that case, some laws are applied on local variables to determine their behavior. Moreover a mapping is required between those local variables and the global coordinates. In Siconos, this is achieved thanks to `NonSmoothLaw` and `Relation` objects, gathered in `Interaction`. A complete review of these classes is done in Section 13.

The bouncing ball is subjected to a simple constraint : its motion is limited by the ground. In other words, this unilateral constraint states that the distance between the ball and the ground must always remain nonnegative. Moreover, we assume that the local velocity after impact is proportional to the velocity before impact, as

$$\dot{y}(t^+) = -e\dot{y}(t^-)$$

y being the distance between the ball and the ground and $t^{\pm}$ post/pre impact time instants. This whole behavior is modelized thanks to a Newton impact law :

$$\text{if } y(t) = 0, \quad 0 \leqslant \dot{y}(t^+) + e\dot{y}(t^-) \perp \lambda \geqslant 0$$

with complementarity between $\dot{y}$ and λ, the Lagrange multiplier associated to the generalized reaction force p. Those local variables are related to global coordinates of the dynamical system in a very simple way:

$$\begin{aligned} y &= z - R - h \\ p &= \lambda \end{aligned}$$

The relation above fits with `LagrangianLinearTIR` (Time Invariant Relation)

$$\begin{aligned} y &= Hq + b, \ H = [1] \ b = -R - h \\ p &= H^t \lambda \end{aligned}$$

Consequently, Equations 1 and 1 are translated into Siconos language, for the bouncing ball, as

```
# The relation, mapping between local and global variables
H = [[1]]
b = -radius
relation = SK.LagrangianLinearTIR(H, b)
# A nonsmooth law to be applied on local variables
e = 0.9 # restitution coefficient
nslaw = SK.NewtonImpactNSL(e)
# An interaction gathers the newton impact law and the Lagrangian linear relation
inter = SK.Interaction(1, nslaw, relation)
# link the interaction to the dynamical system
bouncingBall.nonSmoothDynamicalSystem().link(inter, ball)
```

At this point, the nonsmooth dynamical system is completely defined, gathering the smooth dynamics with a potential contact with the ground. The behavior at contact is defined through the interaction. The next steps consist in the description of the simulation process.

First of all, a time discretization must be defined

```
# start at 0 with a fixed time step of 0.05
tstart = 0.0
time_step = 0.05
td = SK.TimeDiscretisation(tstart, time_step)
```

The choice of the simulation types determines the whole strategy used to formalize and solve the nonsmooth problem. Here, an event-capturing Moreau-Jean time-stepping scheme will be used. All details about the possible strategies are available in Section 10. Simulation process is more or less based on two main objects: `OneStepIntegrators` (OSI), telling how to integrate the dynamics over a time step and `OneStepNSProblem` (OSNS), telling how to formalize and solve

the nonsmooth problem. For the bouncing ball, we choose a Moreau-Jean integrator, with a θ scheme

```
theta = 0.5
# Build the integrator that must be associated to the dynamical system
osi = SK.MoreauJeanOSI(ball, theta)
```

In the Moreau-Jean time-stepping scheme, the unilateral constraints, after being reformulated at the velocity level, lead at each time step to nonsmooth optimization problems. In our case, a linear complementarity problem (LCP) is written:

```
# Choose a nonsmooth formulation
osnspb = SK.LCP()
```

The default solver for LCP is Lemke[4].

Finally, the simulation is built and used to initialized the model to connect the simulation components with the nonsmooth dynamical systems and its interactions, among other things.

```
simulation = SK.TimeStepping(td, osi, osnspb)
bouncingBall.initialize(simulation)
```

From then on, the model is complete and the simulation can start. Resolution of the nonsmooth problem between two time instants ("events") is leaded by the `Simulation` thanks to the following methods:

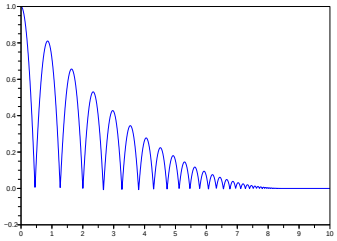
- `simulation.hasNextEvent()` to check if some computation remains to be done.
- `simulation.computeOneStep()` to formalize and solve the nonsmooth problem between two events.
- `simulation.nextStep()` to finalize the process for the current time and prepare the next one.

A typical Siconos time loop looks like:

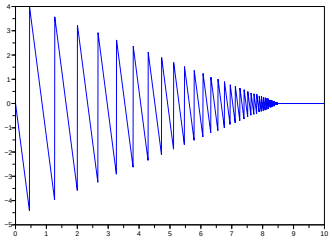
```
while simulation.hasNextEvent():
    simulation.computeOneStep()
    # the current time is simulation.nextTime()
    # the current ball position is ball.q()
    # the current ball velocity is ball.velocity()
    # the current reaction force is ball.p(1)
    simulation.nextStep()
```

The results of the simulation are presented on Figure 2.

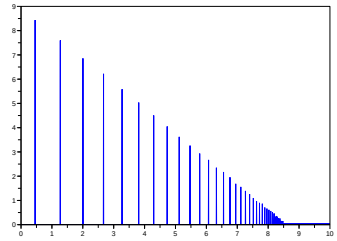
[4]For a detailed list of the available solvers, see `http://siconos.gforge.inria.fr/Numerics/LCProblem.html`



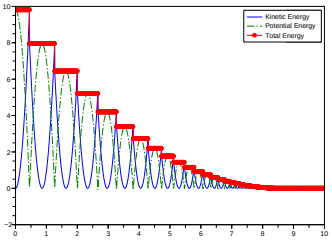
(a) Position of the ball vs. time



(b) Velocity of the ball vs. time



(c) Reaction due to the contact force vs. time



(d) Energy balance vs. time

Figure 2: Simulation results (time-stepping) for a ball bouncing on the ground.

2 A diode bridge

The present example is a four diodes bridge wave rectifier as shown on Figure 3.

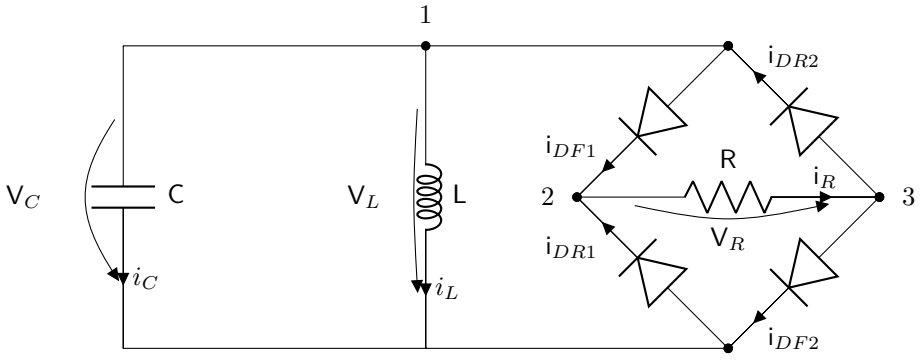


Figure 3: A 4-diodes bridge wave rectifier

A LC oscillator, initialized with a given voltage across the capacitor and a null current through the inductor, provides the energy to a load resistance through a full-wave rectifier consisting of a 4 ideal diodes bridge. Both waves of the oscillating voltage across the LC are provided to the resistor with current flowing always in the same direction. The energy is dissipated into the resistor and results in a damped oscillation.

As for the bouncing ball, the first step is the definition of a leading `Model`

```
import Siconos.Kernel as SK
# The model sets the time range for the simulation
tstart = 0. ; tend = 10.
DiodeBridge = SK.Model(tstart, tend)
```

The oscillator is a time-invariant linear dynamical system, and using the Kirchhoff current and voltage laws and the branch constitutive equations, its dynamics is written as (see Figure 3 for notation)

$$\begin{bmatrix} \dot{v}_L \\ \dot{i}_L \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{C} \\ \frac{1}{L} & 0 \end{bmatrix} \begin{bmatrix} v_L \\ i_L \end{bmatrix} + \begin{bmatrix} 0 & 0 & -\frac{1}{C} & \frac{1}{C} \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -v_{DR1} \\ -v_{DF2} \\ i_{DF1} \\ i_{DR2} \end{bmatrix}. \tag{1}$$

Denoting

$$x = \begin{bmatrix} \dot{v}_L \\ \dot{i}_L \end{bmatrix}, \quad \lambda = \begin{bmatrix} -v_{DR1} \\ -v_{DF2} \\ i_{DF1} \\ i_{DR2} \end{bmatrix}, \quad A = \begin{bmatrix} 0 & \frac{-1}{C} \\ \frac{1}{L} & 0 \end{bmatrix}, \quad r = \begin{bmatrix} 0 & 0 & \frac{-1}{C} & \frac{1}{C} \\ 0 & 0 & 0 & 0 \end{bmatrix} \lambda$$

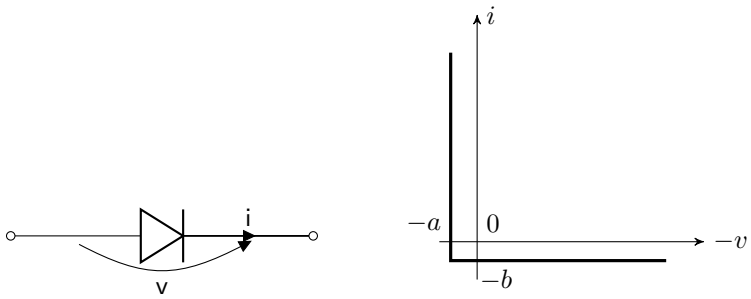
the dynamical system (1) may be rewritten as

$$\dot{x} = Ax + r$$

which fits with `FirstOrderLinearDS` formalism

```
Lvalue = 1e-2     # inductance
Cvalue = 1e-6     # capacitance
Rvalue = 1e3      # resistance
Vinit = 10.0      # initial voltage
init_state = [Vinit, 0]
A = [[0, -1.0/Cvalue], [1.0/Lvalue, 0]]
LSDiodeBridge = SK.FirstOrderLinearDS(init_state, A)
# insert the dynamical system into the model
DiodeBridge.nonSmoothDynamicalSystem().insertDynamicalSystem(LSDiodeBridge)
```

On Figure 4 below, the left-hand sketch displays the ideal diode characteristic while the right-hand sketch displays the usual exponential characteristic as stated by Shockley's law. Thus the behavior



(a) Diode symbol. (b) Ideal diode modelling.

Figure 4: Complementarity modelling of the diode

of each diode of the bridge, supposed to be ideal, can be described with a complementarity condition between current and reverse voltage :

$$\begin{aligned} 0 \leqslant -v_{DR1} \perp i_{DR1} \quad &\geqslant 0 \\ 0 \leqslant -v_{DF2} \perp i_{DF2} \quad &\geqslant 0 \\ 0 \leqslant i_{DF1} \quad \perp -v_{DF1} &\geqslant 0 \\ 0 \leqslant i_{DR2} \quad \perp -v_{DR2} &\geqslant 0 \end{aligned}$$

which is equivalent to

$$0 \leqslant y \perp \lambda \geqslant 0 \; with \; y = \begin{bmatrix} i_{DR1} \\ i_{DF2} \\ -v_{DF1} \\ -v_{DR2} \end{bmatrix}$$

With y and λ as local variables, 2 is a nonsmooth law , a `ComplementarityConditionNSL` indeed, defined is Siconos as :

```
# "4" is the size of the ns law, i.e. the length of y
nslaw = ComplementarityConditionNSL(4)
```

To complete this law and to write a proper interaction, we also need a mapping between local and state variables. To this purpose, a linear relation between voltage and current inside the circuit may be written:

$$\begin{bmatrix} i_{DR1} \\ i_{DF2} \\ -v_{DF1} \\ -v_{DR2} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ -1 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} v_L \\ i_L \end{bmatrix} + \begin{bmatrix} \frac{1}{R} & \frac{1}{R} & -1 & 0 \\ \frac{1}{R} & \frac{1}{R} & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -v_{DR1} \\ -v_{DF2} \\ i_{DF1} \\ i_{DR2} \end{bmatrix}$$

or

$$y = Cx + D\lambda,\ with\ C = \begin{bmatrix} \frac{1}{R} & \frac{1}{R} & -1 & 0 \\ \frac{1}{R} & \frac{1}{R} & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, D = \begin{bmatrix} -v_{DR1} \\ -v_{DF2} \\ i_{DF1} \\ i_{DR2} \end{bmatrix}$$

completed with the relation between r and *lambda* in Equation 2, this corresponds to a `FirstOrderLinearTIR`

```
C = [[0.,   0.], [0,    0.], [-1.,  0.], [1.,   0.]]
D = [[1./Rvalue, 1./Rvalue, -1.,  0.],
    [1./Rvalue, 1./Rvalue,  0., -1.],
    [1.,        0.,         0.,  0.],
    [0.,        1.,         0.,  0.]]
B = [[0.,        0., -1./Cvalue, 1./Cvalue],
     [0.,        0.,  0.,        0.       ]]
LTIRDiodeBridge = FirstOrderLinearTIR(C, B)
LTIRDiodeBridge.setDPtr(D)
```

Everything is now ready to build the whole nonsmooth dynamical system:

```
# First, the interaction
InterDiodeBridge = SK.Interaction(nslaw, LTIRDiodeBridge)
# Then, link this interaction to the dynamical system
DiodeBridge.nonSmoothDynamicalSystem().link(InterDiodeBridge, LSDiodeBridge)
```

At this point, the modeling process is complete and ready for simulation. As for the bouncing ball example, an event-capturing Moreau-Jean time-stepping strategy will be applied. The simulation writing is exactly the same as for the bouncing ball, say:

```
# start at 0 with a fixed time step of 0.05
tstart = 0.0
time_step = 0.05
td = SK.TimeDiscretisation(tstart, time_step)
# Set integrator for the dynamical system
theta = 0.5
```

```
osi = SK.EulerMoreauOSI(LSDiodeBridge, theta)
# Choose a nonsmooth formulation
osnspb = SK.LCP()
# Build the simulation
simulation = SK.TimeStepping(td, osi, osnspb)
# Initialize the model
DiodeBridge.initialize(simulation)
```

The time loop remains the same:

```
x = LSDiodeBridge.x()
lambda = InterDiodeBridge.lambda_(0)
y = InterDiodeBridge.y(0)
while simulation.hasNextEvent():
    simulation.computeOneStep()
    # inductor voltage is x[0]
    # inductor current is x[1]
    # diode R1 current is y[0]
    # diode R1 voltage is lambda[0]
    # diode F2 voltage is lambda[1]
    # diode F1 current is lambda[2]
    # resistor current is y[0] + lambda[2]

    simulation.nextStep()
```

The final results of the previous simulation are presented on Figure 5.

3 Summary

From the two previous examples, one can easily extract the common structure to any Siconos driver

```
# Build a model

# Build some dynamical systems

# Build some interactions, defining the local behavior
# when dynamical systems interact
# --> nonsmooth law between local variables
# --> relations to map local variables to global coordinates

# Build a simulation
# --> OneStepIntegrators : how to integrate the smooth dynamics
# --> OneStepNSProblem : how to formalize and solve the nonsmooth problem
```

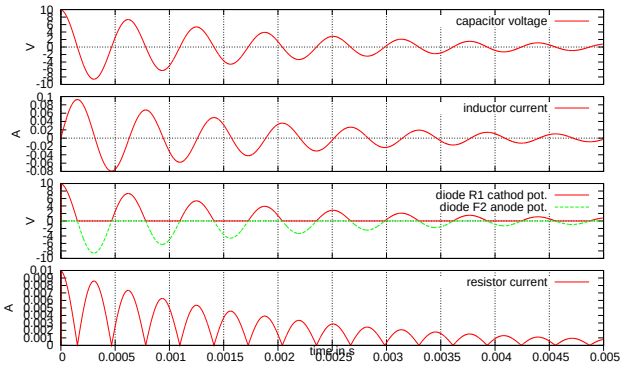


Figure 5: Diodes bridge wave rectifier simulation results.

```
  # Initialize the model

  # Run : time integration of the model.
```

Part II

Overview of NonSmooth Dynamical Systems (NSDS)

NonSmooth Dynamical Systems (NSDS) are dynamical systems characterized by the nonsmoothness of their time evolution and of their formulations. The class of nonsmooth dynamical systems recover a large variety of dynamical systems that arise in many applications. The term "nonsmooth", as for the term "nonlinear", does not define in a precise way the scope of the systems we are interested in. In practice, nonsmooth dynamical systems are defined by their nonsmooth formulations and their nonsmooth time evolution. Most importantly, they share common mathematical and numerical properties. This latter aspect differs from the very general definition of hybrid systems. To be more precise, the nonsmooth analysis of dynamical systems mainly concerns systems that possess the following properties.

(i) A nonsmooth formulation of the constitutive laws that define the system. Famous examples of nonsmooth formulations are piecewise smooth functions, multi–valued mappings, inequality constraints, yielding various definitions of dynamical systems such as piecewise smooth systems, discontinuous ordinary differential equations, complementarity systems, projected dynamical systems, evolution or differential variational inequalities and differential inclusions.

(ii) A concept of solutions which do not consider continuously differentiable functions of time as many times as we want. For instance, simply (absolutely) continuous or Lipschitz functions of time may be solutions of piecewise smooth systems. Measures or distributions are also solutions of interest for differential inclusions.

Other very important features of nonsmooth dynamical systems that will be presented in the sequel are the corpus of mathematical results that they shared and the ability to efficiently simulate them. These two properties are most of the time closely related. Let us give a list of these properties that creates a specific interest for the nonsmooth dynamical systems:

(i) Mathematical concept of solutions: existence and possibly uniqueness property, continuous dependence on initial conditions

(ii) Dynamical properties: existence of invariants (equilibrium, limit cycles, ...) and their stability. Periodic solutions and waves propagation.

(iii) Control theoretic properties: passivity, controllability, observability, robustness

(iv) Numerical time integration methods: convergence, efficiency (order, robustness,)

(v) Numerical solution procedure for the time–discretized problem.

The instants of discontinuity of the state or its derivatives can be viewed as events, or transitions, when the structure of the system is modified. In this way, a NSDS combines features of continuous dynamical systems with the characteristics of finite automata. Thus, as a mixture of time-continuous dynamics and discrete systems, the NSDS can be viewed as a subclass of hybrid systems.

As we said earlier, as the term "hybrid", the "nonsmooth" one is not an accurate definition. A better way to define a coherent class of dynamical systems is to consider the mathematical nature of their solutions depending on the chosen formulation. This introduction aims at giving a flavor of the mathematical properties and of the numerical consequences, that are shared by NSDS. The term "nonsmooth" is partly inherited from the extensive use of a well-recognized mathematical theory: the Nonsmooth Analysis [23]. Since they are, at the same time, more specific that the hybrid dynamical systems and more closely related to physical applications, new mathematical results can be derived and efficient simulation tools can be designed. The role of the Siconos platform is then to take advantage of these properties in order to provide general modeling and simulation tools for nonsmooth dynamical systems.

4 Constrained First Order Dynamics

Let us consider a first very simple example of nonsmooth dynamical system that we aim at simulating in Siconos. It is a scalar affine dynamical system

$$\dot{x}(t) = ax(t) + b, x \in \mathbb{R}, x(t_0) = x_0 > 0$$

where a, b are given constants, that is constrained to evolved in the positive orthant, that is

$$x(t) \geqslant 0.$$

if $a > 0, b > 0$, the solution of the system is trivial and remains naturally in the positive orthant. In the other case, the system may reach the boundary of the admissible domain $C = \{x \geqslant 0\}$. In that case, a Lagrange multiplier λ is added to maintain the state in C by writing

$$\dot{x}(t) = ax(t) + b + \lambda.$$

The constraint (4) is called an unilateral constraint since it constraints the system in a one-sided way. Usually, a relation links the Lagrange multiplier λ and the constraint. Indeed, $x > 0$ implies that $\lambda = 0$ since the dynamics is free to evolve in $\mathbb{R}_+$. If $x = 0$, then the multiplier has to maintain the state in the positive orthant and it can do that only if $\lambda \geqslant 0$. This relation is a complementarity relation that is familiar to people who deals with optimization problems and their associated KKT conditions. It can be written

$$0 \leqslant x \perp \lambda \geqslant 0.$$

The symbol $\perp$ means that $x\,\lambda = 0$. Grouping (4) and (4), we obtain one of the simplest linear complementarity system

$$\begin{cases} \dot{x}(t) = ax(t) + b + \lambda. \\ 0 \leqslant x \perp \lambda \geqslant 0. \end{cases}$$

The aim of Siconos is to propose a large number of numerical solution procedures to solve such problems. Especially, it provides numerical time integration procedures, such as event–driven schemes that take into the account the event when the state reaches the boundary of the admissible domain and also time–stepping procedure where the event is not explicitly located. The Siconos software provides numerical routines to solve the underlying problems that comes from the mathematical programming theory. In this simple example, it is a linear

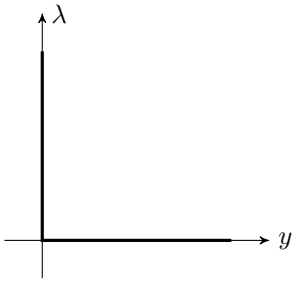


Figure 6: Complementarity condition $0 \leqslant y \perp \lambda \geqslant 0$.

complementarity problem [24] and even better a trivial quadratic program that can be solved by a simple projection [31, 57].

Let us consider a more general system given by a first order dynamical system of the form:

$$\dot{x} = f(x,t), x \in \mathbb{R}^n, t \in [0,T], \text{ with initial condition } x_0 \in \mathbb{R}^n$$

subjected to a set of constraints on its state[5]:

$$y = h(x) = [h_i(x), i = 1 \dots m]^T \geqslant 0. \tag{2}$$

The constraints (2) are usually enforced by an external input, a Lagrange multiplier vector $\lambda \in \mathbb{R}^m$. The latter acts on the dynamical system through an input function $g\colon \mathbb{R}^m \to \mathbb{R}^n$, such that

$$\dot{x} = f(x,t) + \nabla_x^\top h(x)\, \lambda.$$

Finally, in order to complete the description of the system, additional modeling information are required. As before, in simple cases, the relation takes the form of a complementarity condition:

$$0 \leqslant y \perp \lambda \geqslant 0$$

and we get a gradient–type complementarity system

$$\begin{cases} \dot{x} = f(x,t) + \nabla_x^\top h(x)\, \lambda \\ y = h(x) \\ 0 \leqslant y \perp \lambda \geqslant 0 \end{cases}$$

The graph of the complementarity condition is depicted in Figure 6.

The complementarity systems in (4) can be generalized in two directions

a) The complementarity condition can be replaced by a generalized equation (see [62]) between the output y and the multiplier λ, denoted by the following inclusion:

$$0 \in F(y,\lambda) + Q(y,\lambda)$$

where $F\colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is assumed to be continuously differentiable and $Q\colon \mathbb{R}^m \times \mathbb{R}^m \rightrightarrows \mathbb{R}^m$ is a multivalued mapping with a closed graph. Using the notion of normal cone, the

[5]The inequality is to be understood component-wise.

generalized equation appears to be a generalization of the complementarity condition which is in turns a special case of variational inequality[29].

b) The dynamics can be also generalized leading to the definition of Dynamical complementarity systems

$$\left\{\begin{array}{l} \dot{x}(t) = f(t, x(t), \lambda(t)) \\ y(t) = h(t, x(t), \lambda(t)) \\ 0 \leqslant y(t) \perp \lambda(t) \geqslant 0, \end{array}\right.$$

Nature of solutions and relative degree. A fundamental notion that is behind the general definition of dynamical complementarity systems is the notion of relative degree or index. This concept drastically (improve or clarify?) the mathematical nature of the solution and then the numerical methods that we use for simulating. Let us consider the following dynamical system,

$$\left\{\begin{array}{l} x(0) = x_0 \\ x^{(2)}(t) = -1 + \lambda(t) \\ 0 \leqslant x(t) \perp \lambda(t) \geqslant 0 \end{array}\right.$$

The relative degree of (4) is equal to 2. If the initial conditions $x_0 \geqslant 0$ is consistent at the initial time, the solution is sought as an absolutely continuous function, but λ is no longer a function of bounded variations. When the state hits the boundary $x = 0$, λ must contain a Dirac distribution $\delta^{(1)}$ since $\dot{x}$ is not consistent with the constraint. In that case, a reinitialization mapping (in the hybrid system language) or an impact law (in the mechanics language) defining the state of the system after a possible nonsmooth event has to be added:

$$x(t^+) = \mathcal{F}(x(t^-), t)$$

If we consider the following dynamical system,

$$\left\{\begin{array}{l} x(0) = x_0 \\ x^{(3)}(t) = -1 + \lambda(t) \\ 0 \leqslant x(t) \perp \lambda(t) \geqslant 0 \end{array}\right.$$

The relative degree of (4) is equal to 3. If the initial condition $x_0 \geqslant 0$ is consistent at the initial time, the solution is sought as an absolutely continuous function, but λ must contain a derivative of the Dirac distribution $\delta^{(1)}$ and the meaning of the inequality $\lambda(t) \geqslant 0$ has to be rethought. Without going into deeper details, we refer to [8, 5, 1] for the analysis of higher relative degree systems.

5 Dynamical complementarity systems

Let us start with the general definition of dynamical complementarity systems over cones. Let $I \subset \mathbb{R}$ be an interval containing its origin t_0. Let $K \subset \mathbb{R}^m$ be a nonempty closed convex cone and $K^\star$ its dual cone given by

$$K^\star = \{x \in \mathbb{R}^m \mid x^\top y \geqslant 0 \text{ for all } y \in K\}.$$

Definition 1 (Dynamical complementarity systems (DCS) over cones) *Let us consider two smooth ($\mathcal{C}^1$) mappings $f : I \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $h : I \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$. A dynamical*

complementarity system over cones is given as

$$\left\{\begin{array}{l} \dot{x}(t) = f(t, x(t), \lambda(t)) \\ y(t) = h(t, x(t), \lambda(t)) \\ K^\star \ni y(t) \perp \lambda(t) \in K, \end{array}\right.$$

where $t \in I \subset \mathbb{R}$, $x(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^m$ is usually called the output vector.

The notation $y \perp \lambda$ means $y^\top \lambda = 0$. Using basic convex analysis results, standard equivalences

$$K^\star \ni y \perp \lambda \in K \quad \Longleftrightarrow \quad -\lambda \in \mathbb{N}_{K^\star}(y) \quad \Longleftrightarrow \quad -y \in \mathbb{N}_K(\lambda) \quad \Longleftrightarrow \quad -y \in \partial \Phi_K(\lambda),$$

allow one to reformulate (1) into normal cone inclusions (or subdifferential inclusion). Let us recall that the normal cone to K at $x \in K$ is defined by

$$\mathbb{N}_K(x) = \{v \in \mathbb{R}^m | \langle v, y - x \rangle \leqslant 0 \text{ for all } y \in K\}.$$

and the indicator function of a set C

$$\Phi_C(x) = \left\{\begin{array}{ll} 0, & x \in C, \\ +\infty, & x \notin C. \end{array}\right.$$

Finally, the notation $\partial\varphi$ denoted the subgradient of a convex function $\varphi : \mathbb{R}^m \to \mathbb{R}$ and is defined by

$$\partial\varphi(x) = \{v \in \mathbb{R}^m \mid \varphi(y) - \varphi(x) \geqslant \langle v, y - x \rangle \text{ for all } y \in \mathbb{R}^m\}.$$

This reformulation permits to link the theory of dynamical complementarity problem with the theory of differential inclusions.

When the cone $K = \mathbb{R}^m$, we get a very special type of dynamical complementarity system that reduces to the differential algebraic equations

$$\left\{\begin{array}{l} \dot{x}(t) = f(t, x(t), \lambda(t)) \\ 0 = h(t, x(t), \lambda(t)). \end{array}\right.$$

For a detailed presentation and study of differential algebraic equations, we refer to standard textbooks [17, 38, 12]. An important remark is that the notion of index in differential algebraic equations will also have a large influence on the nature of solutions of differential complementarity systems. This point will be detailed in the sequel. We can argue that the differential algebraic equations are not typical nonsmooth dynamical system. As for the ordinary differential equation, one usually seek for smooth solutions and the nonsmoothness appears only in higher derivatives. However, it is noteworthy that inconsistent initial conditions, that is initial conditions that do not respect the constraints $h(t_0, x(t_0), \lambda(t_0)))$ may yield to jump at the initial time.

When the cone K is specialized to the nonnegative orthant of $\mathbb{R}^m$, we get the standard dynamical complementarity problem.

Definition 2 (Dynamical complementarity systems (DCS)) *Let us consider two smooth ($\mathcal{C}^1$) mappings $f : I \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and $h : I \times \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m$. A dynamical complementarity system (DCS) is given as*

$$\left\{\begin{array}{l} \dot{x}(t) = f(t, x(t), \lambda(t)) \\ y(t) = h(t, x(t), \lambda(t)) \\ 0 \leqslant y(t) \perp \lambda(t) \geqslant 0, \end{array}\right.$$

where $t \in I \subset \mathbb{R}$, $x(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^m$.

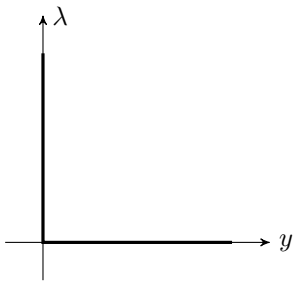


Figure 7: Complementarity condition $0 \leqslant y \perp \lambda \geqslant 0$.

The notation $x \geqslant 0$ holds component–wise. The graph of the complementarity condition is depicted in Figure 7.

If the mapping f and h are affine, we get a Linear Complementarity Systems (LCS).

Definition 3 (Linear complementarity systems (LCS)) *A linear complementarity system (LCS) over cones is given as*

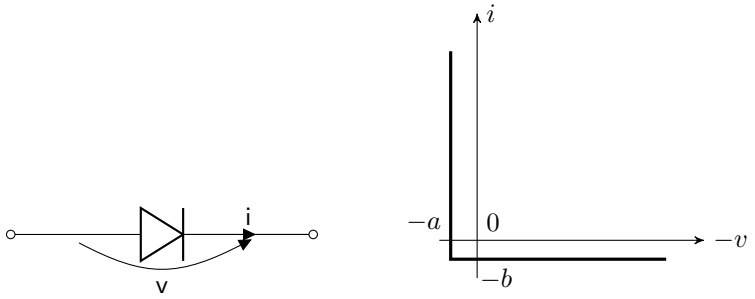
$$\left\{ \begin{array}{l} \dot{x}(t) = Ax(t) + B\lambda(t) + u(t) \\ y(t) = Cx(t) + D\lambda(t) + a(t) \\ K^\star \ni y(t) \perp \lambda(t) \in K, \end{array} \right.$$

where $t \in I \subset \mathbb{R}$, $x(t) \in \mathbb{R}^n$ and $y(t) \in \mathbb{R}^m$ and $A \in \mathbb{R}^{n\times n}, B \in \mathbb{R}^{n\times m}, C \in \mathbb{R}^{m\times n}$ and $D \in \mathbb{R}^{m\times m}$. When $K = \mathbb{R}^m_+$, we simply coin the system (3) a linear complementarity system.

Well posedness properties of LCS have been extensively studied in [22, 39, 21, 20].

XXX say some word ?

Complementarity modelling of electrical components was introduced in the pioneering works of the Eindhoven school led by W.M.G. van Bokhoven [44]. Their aim was to compute the steady state of piecewise linear circuit by means of the complementarity models (see the book [45] for a good account).



(a) Diode symbol. (b) Ideal diode modelling.

Figure 8: Complementarity modelling of the diode with possible residual current b and voltage a

In Figure 8(b), a complementarity modelling of the diode with possible residual current b and

voltage a is depicted. It can be defined by the following complementarity condition, or inclusion into a normal cone as

$$0 \leqslant i+b \perp a-v \geqslant 0 \Longleftrightarrow -(i+b) \in \mathbb{N}_{[-a,+\infty)}(-v).$$

The notation $x \perp y$ means that $x^\top y = 0$. Inequalities involving vectors are understood to hold component-wise. From the definition of the normal cone, it follows that an equivalent definition of the model of the diode as a Variational Inequality(VI) is :

$$(i+b)(v+u) \geqslant 0, \text{ for all } u \in [-a,+\infty).$$

Including such kind of components into circuits composed of inductors, capacitors and resistors yields linear complementarity systems (for more details we also refer to [3]).

In the work of W.M.G. van Bokhoven and co-workers, the notion of solution was extremely clear since it reduces to the question of the existence and possible uniqueness of solutions as a vector, say x, in a finite–dimensional space, say $\mathbb{R}^n$. When we deal with dynamics, the question of the solution as a function of time $x(t) \in \mathbb{R}^n$ is more difficult since we deal with some functional spaces of functions of time. Let us try in the sequel to give some basic arguments for the smoothness of solutions.

6 Relay systems and sliding mode systems

More general complementarity systems may be defined by

$$\left\{\begin{array}{l} \dot{x}(t) = f(t,x(t),\lambda(t)), \\ y(t) = h(t,x(t),\lambda(t)), \\ -y(t) \in \mathbb{N}_X(\lambda(t)), \end{array}\right.$$

where X is a nonempty closed set of $\mathbb{R}^n$. Some instances of the system (6) where X is not cone are also very interesting in practise. Indeed, note that

$$-y(t) \in \mathbb{N}_{[-1,1]}(\lambda(t)) \quad \Longleftrightarrow \quad -\lambda(t) \in \mathrm{Sgn}(y(t)),$$

For a vector $y \in \mathbb{R}^m$, $\mathrm{Sgn}(y)$ holds component-wise. Let us consider for instance that $X = [-1,1]^m$ in (6). We end up with a dynamical relay system

$$\left\{\begin{array}{l} \dot{x}(t) = f(t,x(t),\lambda(t)), \\ y(t) = h(t,x(t),\lambda(t)), \\ -\lambda(t) \in \mathrm{Sgn}(y(t)). \end{array}\right.$$

In the affine case, the well-posedness of linear relay systems has been studied in [61, 6].

In order to say more on the mathematical properties of (6), we note that the inclusion in the third line (6) is equivalent to the following VI

$$y(t)(\tau - \lambda(t)) \geqslant 0, \text{ for all } \tau \in X,$$

that is

$$h(t,x(t),\lambda(t))(\tau - \lambda(t)) \geqslant 0, \text{ for all } \tau \in X.$$

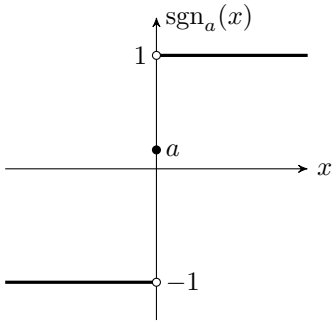


Figure 9: Single–valued sign function

Thanks to (6), the system (6) can be recast into the Differential Variational Inequalities (DVI) framework introduced by [58].As it is noted by the authors, there can be substantial variations of the characteristics of DVIs regarding the existence and regularity of solutions depending mainly on the solution of the VI (6). Let us denote by $\lambda(t) \in \mathrm{SOL}(X, h(t, x(t), \ \cdot\ ))$ an element of $\mathbb{R}^m$ solution of (6). Depending on the mathematical nature of the mapping $(x,t) \mapsto \mathrm{SOL}(X, h(t, x, \ \cdot\ ))$, various types of solutions to (6) are obtained. A brief description is given in the following paragraphs.

7 Discontinuous ordinary differential equations and Filippov solutions.

Let $I \subset \mathbb{R}$ be an interval containing its origin t_0. Let us consider an ordinary differential equation

$$\dot{x}(t) = f(t, x(t)),$$

where $f : I \times \mathbb{R}^n \to \mathbb{R}^n$ is a discontinuous function with respect to x and measurable with respect to t. In that case, the standard and the Carathéodory analysis of ordinary differential equations does no longer apply for all discontinuous functions f. A standard example of such a system is given by the single–valued sign function

$$\mathrm{sgn}_a(x) = \begin{cases} 1, & x > 0 \\ a, & x = 0 \\ -1, & x < 0. \end{cases}$$

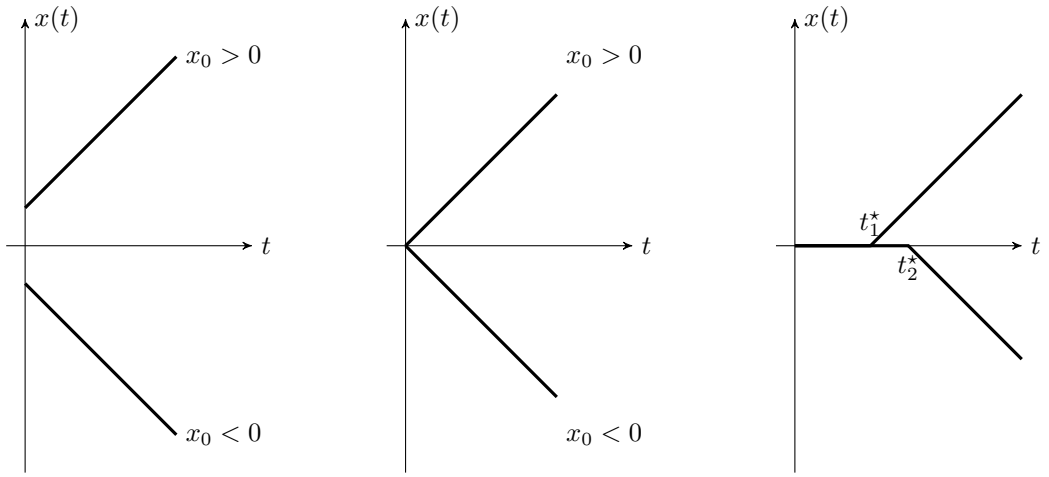
with $a \in \mathbb{R}$ that is depicted in Figure 9. The Cauchy problem, given by

$$\begin{cases} \dot{x}(t) = \mathrm{sgn}_a(x(t)) \\ x(t_0) = x_0 \neq 0, \end{cases}$$

is well defined in the standard way and we get a smooth trivial solution

$$x(t) = x_0 + \mathrm{sgn}_a(x_0)(t - t_0).$$

This solution is illustrated in Figure 10(a). A first issue arises when $x(t_0) = x_0 = 0$. In that case, the problem may have more than one solution in the sense of Carathéodory. If $a \neq 0$, we have two solutions in the sense of Carathéodory, $x(t) = t, t \in I$ and $x(t) = -t, t \in I$ depicted in



(a) $x_0 \neq 0$ (b) $x_0 = 0, a \neq 0$ (c) $x_0 = 0, a = 0$

Figure 10: Multiple Carathéodory solutions of $\dot{x}(t) = \mathrm{sgn}(x(t)$

Figure 10(b). Note that these solutions depend neither on the values of a nor on the sign of a. If $a = 0$, a trivial solution is $x(t) = 0, t \in I$, but an infinite number of solutions are also existing in the sense of Carathéodory. The following functions are indeed Carathéodory solutions for any $t^\star \geqslant t_0$,

$$\begin{cases} x(t) = 0, & t < t^\star \\ x(t) = t, & t \geqslant t^\star \end{cases} \quad \text{or} \quad \begin{cases} x(t) = 0, & t < t^\star \\ x(t) = -t, & t \geqslant t^\star \end{cases}$$

Examples of such solutions are described in Figure 10(c).

Let us consider now the following Cauchy problem given by

$$\begin{cases} \dot{x}(t) = -\,\mathrm{sgn}_a(x(t)) \\ x(t_0) = x_0 \neq 0. \end{cases}$$

It is easy to guess that the trajectory starting from $x_0 \neq 0$ will reach the origin at time $t^\star = t_0 + |x_0|$. After $t^\star$, if $a = 0$, we get $x(t) = 0, t \geqslant t^\star$. If $a \neq 0$, there is no way to define a solution in the sense of Carathéodory. In other words, there is no maximal solution on $[t_0, +\infty]$ for $a \neq 0$.

To remedy to this lack of existence of maximal solution, a first condition, the so-called *transversality condition* is often invoked. Another more general solution is the concept of Filippov solutions and the associated differential inclusion extensions.

Definition 4 (Discontinuous ordinary differential equation. [30]) *Let us consider a piecewise smooth dynamical system as it is defined in Definition* **??** *by a family of smooth mapping $f_i : I \times \bar{X}_i \to \mathbb{R}^n$ continuous up to the boundary over the finite partition $\{X_i\}_{i=1,\ldots,k}$. A discontinuous ordinary differential equation is given by*

$$\dot{x}(t) = f(t, x) = f_i(t, x(t)) \text{ for } x \in X_i, i \in \{1, \ldots k\}.$$

where the vector field $x \mapsto f(x, t)$ is assumed to be discontinuous on a set $M \subset \cup_{i=1}^{k} \partial X_i$ of zero measure. Usually, the set M is given by a finite number of surfaces $S_s, s = 1, \ldots, n_s$ of

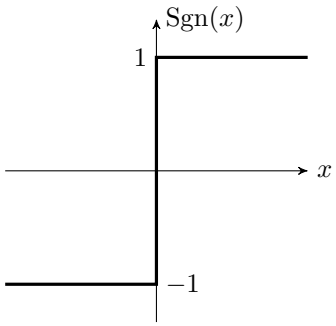


Figure 11: Multivalued–valued sign function

co-dimension d_s given by

$$S_s = \{x \mid \varphi_s(x) = 0, x \in G_s\}, \quad s = 1, \ldots, n_s,$$

where G_s are domains of definition of φ_s and $\varphi_s : G_s \to \mathbb{R}^{d_s}$ are in $\mathcal{C}^1(G_s)$.

Notion of Filippov solutions and the sliding motion [30]. If the transversality condition is not satisfied or if we are not able to apply it, the notion of differential equation with the Carathéodory solution might not sufficient to define a solution. In the Filippov Theory, the differential equation (4) is embedded (or extended) in a differential inclusion

$$\dot{x}(t) \in F(t, x(t)),$$

where $F : \mathbb{R} \times \mathbb{R}^n \rightrightarrows \mathbb{R}^n$ is a set–valued mapping. At a point (t, x) where $f(t, x)$ is continuous, the set $F(t, x)$ coincides with the singleton $\{f(t, x)\}$. If (t, x) belongs to M, the set of discontinuity points, the set $F(t, x)$ is constructed in an alternative way that we will further detail. Let us define now the concept of Filippov solutions.

Definition 5 (Filippov solutions. [30]) *A Filippov solution of the discontinuous ordinary differential equation (4) is an absolutely continuous function $x(t)$ defined on an interval I such that the differential inclusion (7) is satisfied almost everywhere on I.*

Some examples of the Filippov convex extension Let us consider first the example (7) in the light of the simpler convex Filippov extension. The convex extension leads to the following definition of the set-valued sign mapping as

$$\mathrm{Sgn}(x) = \begin{cases} 1, & x > 0 \\ [-1, 1], & x = 0 \\ -1, & x < 0. \end{cases}$$

which is depicted in Figure 11. The differential equation (7) is therefore replaced by

$$\dot{x}(t) \in \mathrm{Sgn}(x(t)).$$

Whatever the value of the single–valued sign function at $x = 0$, we have a unique maximal Filippov solution for $t \in [t_0, +\infty)$.

8 Mechanical systems with constraints, impact and Coulomb friction

Let us end this introductory chapter with one of most important application nonsmooth dynamical systems: mechanical systems with unilateral constraints, friction and impacts . The equations of motion of mechanical systems with unilateral constraints in a pure Lagrangian setting are

$$\begin{cases} q(t_0) = q_0,\, v(t_0) = v_0, \\ \dot{q}(t) = v(t), \\ M(q(t))\dot{v}(t) + F(t, q(t), v(t)) = G(t, q)\lambda(t), \\ g^\alpha(t, q(t)) = 0, \quad \alpha \in \mathcal{E}, \\ g^\alpha(t, q(t)) \geqslant 0, \quad \lambda^\alpha \geqslant 0, \quad \lambda^\alpha g^\alpha(t, q) = 0 \quad \alpha \in \mathcal{I}, \end{cases}$$

where

- $q(t) \in I\!R^n$ is the generalized coordinates vector and $v(t) = \dot{q}(t)$ the associated generalized velocities vector,
- the initial conditions are $q_0 \in I\!R^n$ and $v_0 \in I\!R^n$,
- $M(q(t)) \in I\!R^{n\times n}$ is the inertia, $F(t, q(t), v(t)) \in I\!R^n$ the forces,
- the function $g(t, q(t)) \in I\!R^m$ defines the constraints in the dynamical system, and $G^\top(t, q(t)) = \nabla_q^\top g(t, q(t))$ is the Jacobian matrix of g with respect to q,
- $\lambda \in I\!R^m$ is the Lagrange multiplier vector associated with the constraints, and
- the sets $\mathcal{E} \subset I\!N$ and $\mathcal{I} \subset I\!N$ respectively describe the set of bilateral constraints (joints) and unilateral constraints (contacts).

In the Newton/Euler formalism [37, 33, 16], the vector of parameters q usually contains the position of the center of mass x and a parametrization of the finite rotation θ which models the orientation of the body with respect to a spatial frame. The velocity is usually composed of the velocity of the center of mass $\dot{x}$ and of an angular velocity Ω expressed for instance in the inertial frame. Therefore, the velocity is not the time–derivative of the parameter vector q, but generally related to q by means of an operator $T(q)$ such that

$$\dot{q}(t) = T^\top(q(t))v(t). \tag{4}$$

The equations of motion (3) can be extended to the Newton/Euler formalism by considering (4) rather than (3b) and by defining G as

$$G(t, q(t)) = \nabla_q g(t, q(t))\, T(q((t)).$$

Remark 1 *After a space–discretization of continuum solids by a finite element approach, the generalized coordinates vector q usually contains the nodal displacements, and possibly the nodal rotations if any. Nevertheless, the generalized velocity is most of the time–derivative of the coordinates q.*

For the sake of simplicity, we also restrict our presentation to holonomic perfect unilateral constraints, that is, we will consider in this paper that $\mathcal{E} = \emptyset$ and that the constraints are scleronomic constraints, *i.e.* $g(t, q(t)) = g(q(t))$. Applications in Part V will however show more general cases. The constitutive law for the perfect unilateral constraints is given by the Signorini condition

$$0 \leqslant g(q(t)) \perp \lambda(t) \geqslant 0,$$

where the inequalities involving vectors are understood to hold component-wise and the $x \perp y$ symbol means that $y^\top x = 0$. Let us define the local velocity $U(t)$ and the generalized reaction forces $r(t)$ which is associated with the (local) Lagrange multiplier $\lambda(t)$ such that

$$U(t) = G^\top(q)\, v(t), \qquad r(t) = G(q)\lambda(t).$$

For finite-freedom mechanical systems, an impact law must be added to close the system of equations. The most simple impact law is the Newton impact law

$$U^+(t) = -e\, U^-(t), \text{ if } g(q(t)) = 0,$$

where e is the coefficient of restitution. For a thorough presentation of enhanced impact laws, we refer to [56].

To illustrate the formulation of nonsmooth mechanical systems, the following academic test examples are archetype that are usually invoked.

Example 1 (The linear oscillator with a stop) *The dynamics of this one-degree-of-freedom system depicted in Figure 12(b) example is a linear spring–damper oscillator, that is*

$$\begin{cases} m\dot{v}(t) + cv(t) + kq(t) = \lambda(t), & \dot{q}(t) = v(t), \\ 0 \leqslant q(t) \perp \lambda(t) \geqslant 0, & v^+(t) = -ev^-(t), \text{ if } q(t) = 0, \end{cases}$$

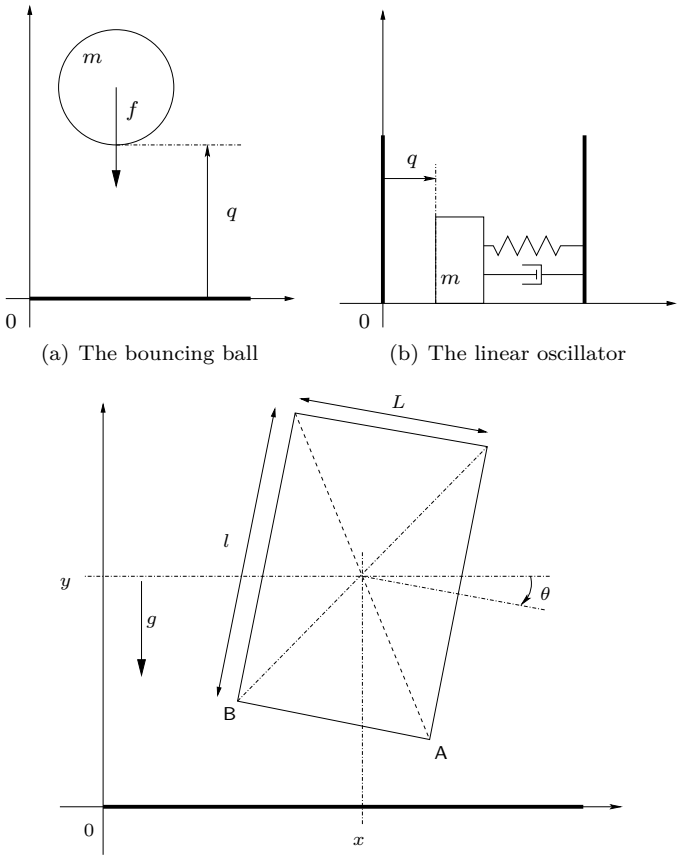
The explicit analytical solution with impacts can be found in [42]. An illustration of this solution is given in Figure 13(a). One of the characteristics of this simple system is that the dynamics is fully linear between two impacts. The impacts occur every half–period of the linear oscillator and there is no accumulation of impacts in time.

Example 2 (The bouncing ball) *This is the standard bouncing ball under gravity depicted in Figure 12(a). The dynamics is constant with a forcing term equal to f together with a unilateral contact on the ground,*

$$\begin{cases} \dot{v}(t) = f(t) + \lambda(t), & \dot{q}(t) = v(t), \\ 0 \leqslant q(t) \perp \lambda(t) \geqslant 0, & v^+(t) = -ev^-(t), \text{ if } q(t) = 0, \end{cases}$$

The interesting feature of the bouncing ball example is the presence of a finite accumulation of impact when $0 < e < 1$ and $f < 0$. The analytical solution of this example can be found in [18]. A more pleasant analytical solution due to Ballard [14] for $f = -2$ and $e = 1/2$ is detailed in the sequel.

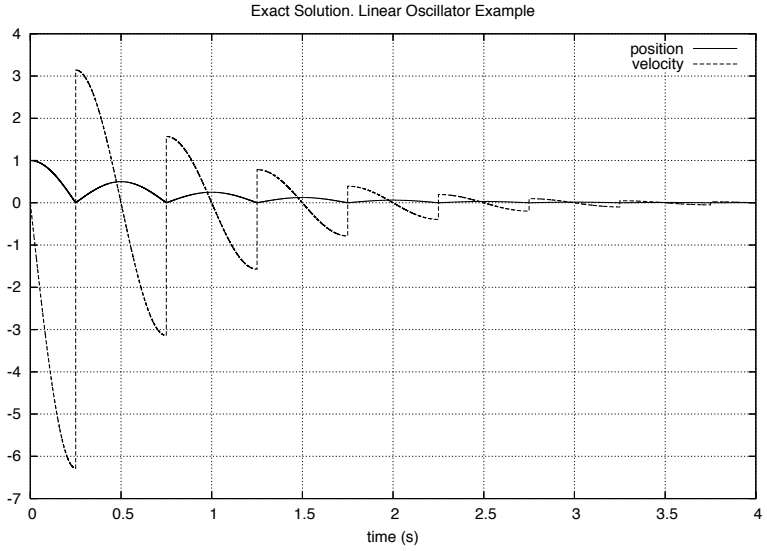
The interesting feature of the bouncing ball example is the presence of a finite accumulation of impact when $0 < e < 1$ and $f < 0$. The analytical solution of this example can be found in [18]. A more pleasant analytical solution due to Ballard [14] is provided in the sequel. It will be used



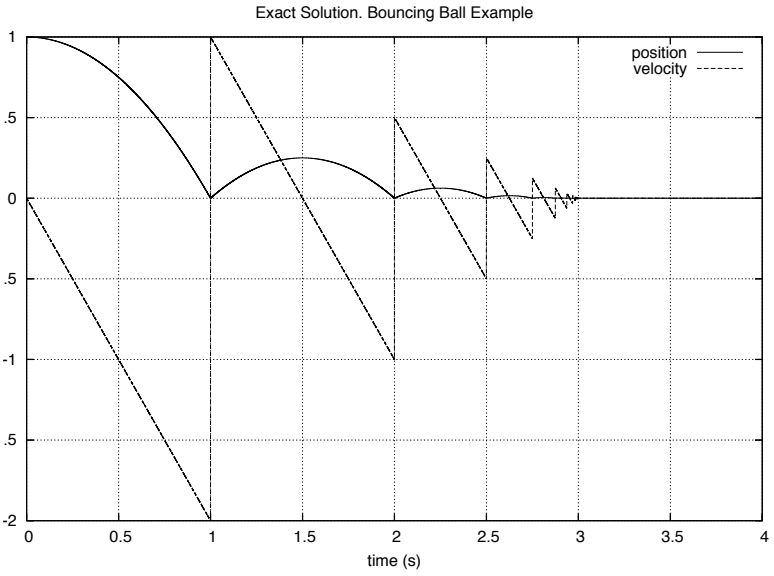
(a) The bouncing ball (b) The linear oscillator

(c) The rocking block

Figure 12: Simple archetypal test examples.



(a) Exact solution for the linear oscillator



(b) Exact Solution for the bouncing ball

Figure 13: Exact Solution for simple archetypal test examples.

as a benchmark in the further sections. The parameters are chosen as $f = -2$, $e = 1/2$ *and the initial data as* $t_0 = 0, q_0 = 1$ *and* $v_0 = 0$. *The analytical solution reads*

$$\begin{cases} q(t) = -t^2 + 1, \\ v(t) = -2t, \end{cases} \qquad t \in [0, 1)$$

$$\begin{cases} q(t) = -(t-3)^2 - \dfrac{3}{2^n}(t-1) + \dfrac{1}{2^{n-1}}\left(3 - \dfrac{1}{2^n}\right), \\ v(t) = -2(t-3) - \dfrac{3}{2^n}, \end{cases} \qquad t \in \left[3 - \frac{1}{2^{n-1}}, 3 - \frac{1}{2^n}\right),$$

$$\begin{cases} q(t) = 0, \\ v(t) = 0, \end{cases} \qquad t \in [3, +\infty)\,.$$

This solution is depicted in Figure 13(b) where the accumulation of events is clearly observed for $t = 3$.

Example 3 (The rocking block) *The rocking block of length* L *and thickness* l *is depicted in Figure 12(c). Let us consider that the contact with the rigid ground can occur at the corner* A *and at the corner* B. *The block is parametrized by the coordinates of the center of mass* $[x, y]$ *and the angle with respect to the ground* θ, *that is* $q = [x, y, \theta]^\top$. *The unilateral constraints read as*

$$\begin{cases} f_{\mathsf{A}}(q) = y - \frac{l}{2}\cos\theta + \frac{L}{2}\sin\theta \geqslant 0, & \text{for the contact point } \mathsf{A}, \\ f_{\mathsf{B}}(q) = y - \frac{l}{2}\cos\theta - \frac{L}{2}\sin\theta \geqslant 0, & \text{for the contact point } \mathsf{B}. \end{cases}$$

The equations of motion in the frictionless case are

$$\begin{cases} m\ddot{x} = 0 \\ m\ddot{y} = -mg + \lambda_{\mathsf{A}} + \lambda_{\mathsf{B}} \\ I\ddot{\theta} = \lambda_{\mathsf{A}}[\frac{l}{2}\sin\theta + \frac{L}{2}\cos\theta] + \lambda_{\mathsf{B}}[\frac{l}{2}\sin\theta - \frac{L}{2}\cos\theta] \end{cases}$$

where m *is the mass of the block and* $I = \frac{m}{12}(l^2 + L^2)$ *the inertia. Despite the fact that the Newton impact law might not be the most appropriate law for reproducing the rocking behaviour of the block, we have chosen this example for the strong coupling between the contact points and the nonlinear constraints. Especially, the projection onto the constraints of one of the contact points can lead to a violation of the constraint for the other contact point if it has not been taken into account in a proper way.*

Perfect bilateral constraints as a normal cone inclusion. Let us consider a set of μ perfect bilateral constraints on the generalized coordinates:

$$g^\alpha(t, q(t)) = 0, \quad \alpha \in \mathcal{E}$$

where the functions $g^\alpha(\,\cdot\,)$ are sufficiently smooth with regular gradients, $\nabla_q g^\alpha(\,\cdot\,,\,\cdot\,)$. The function $g^{\mathcal{E}} : \mathbb{R}^n \times [0, T] \to \mathbb{R}^\mu$ is defined as the vector collecting the functions $g^\alpha(\,\cdot\,), \alpha \in \mathcal{E}$,

$$g^{\mathcal{E}}(t, q) = [g^\alpha(t, q(t)), \alpha \in \mathcal{E}]^\top$$

The bilateral constraints define the configuration manifold $\mathsf{M}(t)$, in which the system must evolve:

$$\mathsf{M}(t) = \left\{ q(t) \in \mathbb{R}^n \;\middle|\; g^{\mathcal{E}}(t, q(t)) = 0 \right\}$$

These bilateral constraints are usually enforced by a set of Lagrange multipliers, $\lambda^\alpha \in I\!R^\mu, \alpha \in \mathcal{E}$. Therefore, the equations of motion are given by:

$$M(q(t))\frac{dv}{dt}(t) + F(t, q(t), v(t)) = \nabla_q g^{\mathcal{E}}(t, q(t))\,\lambda^{\mathcal{E}}$$

where the terms $\nabla_q g^{\mathcal{E}}(t, q)\,\lambda^{\mathcal{E}}$ represent the generalized forces or generalized reactions due to the constraints.

This description of holonomic bilateral constraints can be a little generalized by introducing the tangent space to the manifold M at q

$$T_{\mathsf{M}}(q) = \{\xi \in I\!R^n \;\mid\; \nabla_q^\top g^{\mathcal{E}}(t, q)\xi = 0\}$$

and the normal space as the orthogonal to the tangent space[6]

$$\mathbb{N}_{\mathsf{M}}(q) = \{\eta \in I\!R^n \;\mid\; \eta^\top \xi = 0, \forall \xi \in T_{\mathsf{M}}\}$$

It is noteworthy that the linearly independent rows of the gradient $\nabla_q g^{\mathcal{E}}(t, q)$ form a basis of $\mathbb{N}_{\mathsf{M}}(q)$. The bilateral holonomic constraints are said to be perfect if the multipliers $\lambda^{\mathcal{E}}$ satisfy the following inclusion:

$$r = \nabla_q g^{\mathcal{E}}(q, t)\,\lambda^{\mathcal{E}} \in \mathbb{N}_{\mathsf{M}}(q)$$

We will see in the sequel that this formulation in terms of an inclusion is very useful in practice. We will also omit the term $\nabla_q g^{\mathcal{E}}(t, q)\,\lambda^{\mathcal{E}}$ that corresponds to the bilateral constraints for the sake of simplicity and because the main concern of this book is about unilateral constraints.

Remark 2 *One usually writes $r = -\nabla_q g^{\mathcal{E}}(t, q)\,\lambda^{\mathcal{E}}$ so that all the gradients that enter the dynamics have the same sign. In the bilateral case since the multiplier λ is not signed this is not important.*

Perfect unilateral constraints as a normal cone inclusion. In the Lagrangian setting, the unilateral constraints are usually described by a set of ν inequalities,

$$g^\alpha(t, q) \geqslant 0, \quad \alpha \in \mathcal{I}$$

where the functions $g^\alpha(\,\cdot\,)$ are assumed to be sufficiently smooth with regular gradients. The function $g_{\mathcal{I}} : [0, T] \times I\!R^n \to I\!R^\nu$ is defined as the vector collecting the functions $g^\alpha(\,\cdot\,)$,

$$g_{\mathcal{I}}(t, q) = [g^\alpha(t, q), \alpha \in \mathcal{I}]^\top$$

These unilateral constraints define the subset $\mathsf{U}(t)$ of the configuration space where the system is constrained to evolve:

$$\mathsf{U}(t) = \{q \in I\!R^n \;\mid\; g^\alpha(t, q) \geqslant 0, \alpha \in \mathcal{I}\}$$

As for the bilateral constraints, the unilateral constraints are enforced in the equations of motion by a set of Lagrange multipliers $\lambda^{\mathcal{I}} \in I\!R^\nu$ such that the equation of motion is given by:

$$M(q(t))\frac{dv}{dt}(t) + F(t, q(t), v(t)) = \nabla_q g_{\mathcal{I}}(t, q(t))\,\lambda^{\mathcal{I}}.$$

[6] A metric based on the mass matrix is also habitually used.

The vector $n_\alpha(t,q) = \nabla_q g^\alpha(t,q)$ is a normal vector (not necessarily unit) to the surface $\partial\mathsf{U}(t)$ directed toward the admissible region $\mathsf{U}(t)$.

In a perfect unilateral constraint setting, it is assumed that the reaction force lies along the normal vectors. Finally, when the function $g^\alpha(\,\cdot\,,\,\cdot\,)$, is positive, the corresponding reaction force must be zero, which leads to the following complementarity condition (the so-called Signorini condition):

$$g^\alpha(t,q) \geqslant 0, \quad \lambda_\alpha \geqslant 0, \quad \lambda_\alpha\, g^\alpha(t,q) = 0, \quad \alpha \in \mathcal{I}$$

which will be denoted as before as:

$$0 \leqslant g_{\mathcal{I}}(t,q) \perp \lambda^{\mathcal{I}} \geqslant 0$$

The vector inequalities in (8) have to be understood componentwise.

In a more general way, the outward normal cone to the set $\mathsf{U}(t)$ is defined as:

$$\begin{aligned}\mathbb{N}_{\mathsf{U}(t)}(q(t)) \quad &= \{y \in I\!R^n \;\mid\; y = -\textstyle\sum_\alpha \lambda^\alpha \nabla g^\alpha(q,t),\ \lambda^\alpha \geqslant 0, \text{ for all } \alpha \text{ such that}\\ &\quad\ g^\alpha(q,t) = 0\}\end{aligned}$$

Defining the generalized force $p \in I\!R^n$ corresponding to the the unilateral constraints as

$$p = \sum_\alpha \nabla_q g^\alpha(q,t)\,\lambda^\alpha$$

or more compactly as

$$p = \nabla_q g(q,t)\,\lambda$$

the complementarity condition can be formulated as an inclusion into the normal cone:

$$-p \in \mathbb{N}_{\mathsf{U}(t)}(q(t))$$

Remark 3 *Under the constraint qualification: for all $x \in \mathsf{U}(t)$, there exists $d \in I\!R^n$ such that $\nabla g^{\alpha,T}(q,t)d > 0$ for all α such that $g^\alpha(q,t) = 0$, then the normal cone in (8) and the normal cone of convex analysis $\mathbb{N}_{U(t)}(q(t)) = \{s \in I\!R^n \;\mid\; s^T(y - q(t)) \leqslant 0$ for all $y \in \mathsf{U}(t)\}$ are equal.*

Second order dynamics as a normal cone inclusion Using (8) and (8), the dynamics (3) can be reformulated as a differential inclusion as

$$-(M(q(t))\frac{dv}{dt}(t) + F(t,q(t),v(t))) \in \mathbb{N}_{\mathsf{C}(t)}(q(t))$$

where

$$\mathsf{C}(t) = \mathsf{M}(t) \cap \mathsf{U}(t) = \{q \in I\!R^n \;\mid\; g^\alpha(t,q) = 0, \alpha \in \mathcal{E}, g^\alpha(t,q) \geqslant 0, \alpha \in \mathcal{I}\}$$

A huge amount of work has been published in the literature on DIs, but this kind of inclusion is very particular for two main reasons:

- The right-hand-side is neither bounded and then nor compact. This yields a UDI.
- The inclusion and the constraints concern the second-order time-derivative of q, i.e., the acceleration. This fact leads to strong difficulties, and consequently tools for UDI based on monotone set-valued operator and first-order sweeping process cannot be used.

Such kind of inclusions yields in most of the cases a nonsmooth evolution where the velocity may have jumps, and therefore the acceleration cannot be defined in the usual sense. In the sequel, we will describe briefly some works that tackle the nonsmooth problem as a whole, i.e. a UDI on the second order derivative with a nonsmooth evolution as it has been developed in [64, 54, 51, 43, 52].

Measure Differential Inclusions With the presence of the unilateral constraints, the evolution of the systems is usually no longer smooth. Especially, the velocity $v(\ \cdot\ ) = \dot{q}(\ \cdot\ )$ may encounter jumps and must be considered as a function of bounded variations (BV) in time. With this assumption, the equation of motion is rewritten in terms of right continuous BV (RCBV) function, denoted as $v^+(\ \cdot\ ) = \dot{q}^+(\ \cdot\ )$ [7].

The generalized coordinates, assumed to be absolutely continuous, are deduced from the velocity by the standard integration of a function of bounded variations:

$$q(t) = q(t_0) + \int_{t_0}^{t} v^+(t)\,\mathrm{d}t$$

where $\mathrm{d}t$ is the Lebesgue measure.

If the velocity is a BV function, the acceleration is no longer defined everywhere as the derivative in the classical sense of the velocity. The notion of differential measure, or a special Stieltjes measure provides the right substitute to this notion as a derivative of the velocity in the sense of the distributions. In the same way, the generalized force r is to be considered as a real-measure, denoted di.

The equation of motion (8) is formulated in terms of a measure differential equation:

$$\begin{cases} M(q(t))dv + F_{\mathrm{gyr}}(q(t), v^+(t))\mathrm{d}t + F_{\mathrm{int}}(t, q(t), v^+(t))\mathrm{d}t = F_{\mathrm{ext}}(t)\mathrm{d}t + \mathrm{d}i \\ \\ v^+(t) = \dot{q}^+(t) \end{cases}$$

on $[0, T]$, and with admissible initial data.

Remark 4 *Notice that the dynamics is written in terms of the RCBV function $v^+(\ \cdot\ )$. Is may also possible to write the dynamics in terms of left continuous BV function $v^-(\ \cdot\ )$, as*

$$\begin{cases} M(q(t))dv + F_{\mathrm{gyr}}(q(t), v^-(t))\mathrm{d}t + F_{\mathrm{int}}(t, q(t), v^-(t))\mathrm{d}t = F_{\mathrm{ext}}(t)\mathrm{d}t + \mathrm{d}i \\ \\ v^-(t) = \dot{q}^-(t) \end{cases}$$

on $[0, T]$. Since we are interested only in forward integration of the dynamics, we keep only the form (8).

Decomposition of the Nonsmooth Dynamics Thanks to the Lebesgue decomposition theorem and its variants, the differential measure dv is decomposed as

$$\mathrm{d}v = \gamma\mathrm{d}t + (v^+ - v^-)\mathrm{d}\nu + \mathrm{d}v_s$$

where

[7] Functions of bounded variations always possess right and left limits.

- $\gamma(\ \cdot\ ) = \ddot{q}(\ \cdot\ )$ is the acceleration defined in the usual sense,
- $v^+ - v^-$ is the difference between the right-continuous and the left-continuous functions associated with the BV function $v(\ \cdot\ ) = \dot{q}(\ \cdot\ )$, and $\mathrm{d}\nu$ is a purely atomic measure with atoms at the time t_i of discontinuities of $v(\ \cdot\ )$, i.e.

$$\mathrm{d}\nu = \sum_i \delta_{t_i}$$

- $\mathrm{d}v_s$ is a singular measure with respect to $\mathrm{d}t + d\nu$ which we will neglect for practical reasons.

In the same way, the measure di can be decomposed as follows:

$$\mathrm{d}i = f\mathrm{d}t + p\mathrm{d}\nu + \mathrm{d}r_s$$

where:

- $f(\ \cdot\ )$ is the Lebesgue measurable force,
- p is the purely atomic impact impulsion such that

$$p\mathrm{d}\nu = \sum_i p_i \delta_{t_i}$$

- $\mathrm{d}r_s$ is a singular force measure with respect to $\mathrm{d}t + \mathrm{d}\nu$ which we will also neglect.!

XXX huge problem of consistency in notation with p

The Impact Equations and the Smooth Dynamics Inserting (8) and (8) in (8), the dynamics is written as an equality of measures

$$M(q(t))\gamma(t)\mathrm{d}t + M(q(t))(v^+(t) - v^-(t))\mathrm{d}\nu + F_{\mathrm{gyr}}(q(t), v^+(t))\mathrm{d}t +$$

$$+F_{\mathrm{int}}(t, q(t), v(t))\mathrm{d}t = F_{\mathrm{ext}}(t)\mathrm{d}t + f(t)\mathrm{d}t + p\mathrm{d}\nu$$

and can be split into the atomic part and the Lebesgue part in terms of $v^+(\ \cdot\ )$:

$$\begin{cases} M(q(t))(v^+(t) - v^-(t))\mathrm{d}\nu = p\mathrm{d}\nu \\ x \\ M(q(t))\gamma(t)\mathrm{d}t + F_{\mathrm{gyr}}(q(t), v^+(t))\mathrm{d}t + F_{\mathrm{int}}(t, q(t), v(t))\mathrm{d}t = F_{\mathrm{ext}}(t)\mathrm{d}t + f(t)\mathrm{d}t \end{cases}$$

It is supposed that the unilateral constraints are $g(q) \geqslant 0$, see (8) (8). Due to the definition (8) of the measure $\mathrm{d}\nu$, the impact equations can be written at the time t_i of discontinuities:

$$M(q(t_i))(v^+(t_i) - v^-(t_i)) = p_i$$

This is an algebraic equation. The smooth dynamics which is valid almost everywhere for the Lebesgue measure dt (dt-a.e.) is governed by the following equation:

$$M(q(t))\gamma^+(t) + F_{\mathrm{gyr}}(q(t), v^+(t)) + F_{\mathrm{int}}(t, q(t), v^+(t)) = F_{\mathrm{ext}}(t) + f^+(t)$$

d$t-$a.e., where we assume that $f^+(\ \cdot\ ) = f^-(\ \cdot\ ) = f(\ \cdot\ )\,(\mathrm{d}t - a.e.)$. Obviously the same type of separation between smooth and nonsmooth motions can be performed with the Newton-Euler's equations. The impact dynamics then links the jump in the center of mass velocity and the impulsive contact force, and the instantaneous angular velocity jump with the impulsive contact reaction moment.

Moreau's Sweeping Process Moreau's sweeping process is a mathematical setting which combines a dynamics described in terms of measure as in (8) together with a description of the unilateral constraint including an impact law. We already described quickly the sweeping process in sections **??** and **??**. A key stone of this formulation is the inclusion in terms of velocity. Indeed, the inclusion (8) is "replaced" by

$$-\mathrm{d}i \in N_{T_{\mathcal{C}}(q(t))}(v^+(t))$$

where $\mathcal{C}$ is the admissible domain of the configuration space. We do not make any assumption on $\mathcal{C}$ here, but one should keep in mind that the right-hand-side of (8) may be meaningless for some too general sets $\mathcal{C}$. In most of the cases with practical interest, $\mathcal{C}$ is finitely represented, i.e. it is represented as in (8). In such a case one just as to take care that $\mathrm{Int}(T_{\mathcal{C}}(q)) \neq \emptyset$, which is equivalent to the existence of a hyperplane in $\mathbb{R}^n$, not containing the origin, which intersects all the half-lines generated by the gradients $\nabla g^\alpha(q)$ of the active constraints [53]. This inclusion will be called the inclusion in terms of velocity. Two features of (8) have to be mentioned:

- *The inclusion concerns measures.* Therefore, it is necessary to define what is the inclusion of a measure into a cone.
- *The inclusion is written in terms of velocity* $v^+(\ \cdot\ )$ rather than of the coordinates $q(\ \cdot\ )$.

As we can define an inequality constraint on a measure, it is possible to define a relevant meaning for the inclusion (8). Roughly speaking, when the measure possesses a density with the respect to the Lebesgue measure,

$$\mathrm{d}i = i'\mathrm{d}t = f(t)\mathrm{d}t$$

Then the inclusion is equivalent to the inclusion of $f(\ \cdot\ )$ which is a real function of time, into the cone at time t. When the measure possesses an atom

$$\mathrm{d}i = p\delta$$

where δ is the Dirac measure and p the amplitude of the atom usually called the percussion, the inclusion is equivalent to say that p is included into the cone. Naturally, the same illustration can be made for inequality constraints on measures. For more details, we refer to [51, 43, 65, 8].

A viability lemma due to [55] ensures that the inclusion in terms of velocity (8) together with admissible initial conditions on the position implies that the constraints on the coordinates are always satisfied. In fact, we always have (see e.g. [19] for a proof)

$$N_{T_C(q)}(v^+) \subset N_{\mathcal{C}}(q).$$

The reverse is not true. A key assumption has to be added which is related to the notion of impact laws. Indeed, if the constraint is active, i.e. $dr > 0$, then the post-impact velocity $v^+(\ \cdot\ )$

is equal to zero. For instance if an impact occurs, the post impact velocity vanishes. The model is an inelastic (plastic) impact rule.

As done in [54, 49], the impact rule can be enhanced with normal and tangential coefficients as follows

$$-di \in N_{T_{\mathcal{C}}(q(t))}\left(\frac{v^+(t)+ev^-(t)}{1+e}\right)$$

Inserting (8) in (8) and using (**??**) one obtains

$$v^+(t) = -ev^-(t) + (1+e)\mathrm{prox}_{M(q(t))}[T_{\mathcal{C}}(q(t)); v^-(t)]$$

with $\mathrm{prox}_{M(q(t))}[T_{\mathcal{C}}(q(t)); v^-(t)] = \mathrm{argmin}_{z\in T_C(q(t))}\ \frac{1}{2}(z-v^-(t))^T M(q(t))(z-v^-(t))$, that is numerically tractable since $T_{\mathcal{C}}(q)$ is a polyhedral set. Let $\nu = 1$, i.e. there is only one constraint. This may also be written after some calculations as (q stands for $q(t)$)

$$v^+(t) = v^-(t) - (1+e)M^{-1}(q)\nabla g(q)[\nabla g^T(q)M^{-1}(q)\nabla g(q)]^{-1}\nabla g^T(q)v^-(t)$$

where the multiplier is given by

$$\lambda = -(1+e)[\nabla g^T(q)M^{-1}(q)\nabla g(q)]^{-1}\nabla g^T(q)v^-(t)$$

and $p_i = \nabla g(q)\lambda$ in (8). One may also obtain (8) directly from (8) from the expression of the projection on the tangent cone. If the local relative velocity satisfies $U_{\mathrm{N}}^+(t) = -eU_{\mathrm{N}}^-(t)$ and $U_{\mathrm{T}}^+(t) = U_{\mathrm{T}}^-(t)$ (the case of a frictionless surface), and if $U_{\mathrm{N}}(\ \cdot\ ) = \dot{g}(\ \cdot\ ) = \nabla g^T(q)v(\ \cdot\ )$, then (8) is a consequence of the impact dynamics. Moreau's rule is equivalent to Newton's impact rule, however it is formulated in generalized coordinates and supplies the whole velocity in one shot. When $\nu \geqslant 2$, multiple impacts may occur when the trajectory hits several constraint boundaries at the same time. Moreau's rule also provides a result for the post-impact velocity in this case (notice that (8) is written without assuming that $\nu = 1$). Whether or not the obtained solution is physically sound is another problem. The modelling of multiple impacts is a topic still under investigations at the time of writing of this book. We just mention the fact that Moreau's sweeping process furnishes a geometrical framework that may be used for further research in the field of multiple impacts, and refer to [36, 4] for more information. Moreau's rule in (8) generalizes Newton's law. In [60] it is proposed to extend Poisson's model, sometimes called the kinetic model. This is done by solving two LCPs, one corresponding to the compression phase, the other one to the expansion phase (despite in rigid body theory there are no deformations, so this is to be understood as some kind of approximation of the compliant case).

Finitely Represented $\mathcal{C}$ and the Complementarity Formulation Let $\mathcal{C}$ be finitely represented, i.e.

$$\mathcal{C} = \{q \in \mathcal{M}(t)\ \mid\ g^\alpha(q) \geqslant 0, \alpha \in \{1\dots\nu\}\}$$

In this case the tangent cone is a convex polyhedral set defined by Moreau as

$$T_{\mathcal{C}}(q) = \{z \in I\!R^n\ \mid\ z^T\nabla g^\alpha(q) \geqslant 0, \text{ for all } \alpha \in I(q)\}$$

where $I(q)$ is the set of indices of the active constraints, i.e. $I(q) = \{\alpha \in \{1, .., \nu\} \mid g^\alpha(q) \leqslant 0\}$ [8]. Then we can decompose the measure dr and the velocity $V^+(\ \cdot\ ) = \nabla g^T(q) v^+(\ \cdot\ )$ as follows:

$$\begin{aligned} dr = & \quad \sum_\alpha \nabla g^\alpha(q)\, \mathrm{d}\lambda_\alpha \\ V^+ = & \quad \left[V_\alpha^+ = \nabla g^{\alpha,T}(q)\, v^+, \alpha \in \{1 \dots \nu\}\right] \end{aligned}$$

If some constraints qualification condition holds, then the inclusion (8) can be written equivalently as

$$-\mathrm{d}\lambda_\alpha \in N_{T_{\mathbb{R}_+}(g^\alpha(q))}(V_\alpha^+),$$

or

$$\begin{cases} \text{If } g^\alpha(q) \leqslant 0, \text{ then } 0 \leqslant V_\alpha^+ \perp \mathrm{d}\lambda_\alpha \geqslant 0 \\ \\ \text{If } g^\alpha(q) > 0, \text{ then } \mathrm{d}\lambda_\alpha = 0 \end{cases}$$

This corresponds to a plastic impact ($e = 0$). Replacing V_α^+ by $V_\alpha^+ + eV_\alpha^-$ in (8) and (8) allows one to take into account other restitution with $e \in [0, 1]$. From Claim 6.1 in [18] this is equivalent to formulate the impact at the generalized velocity level as in (8).

Remark 5 *We have not written U^α for the velocity because the term $\nabla g^{\alpha,T}(q)\, v^+$ does not necessarily represent the local kinematics variable as in Section* **??**. *It does for a particular choice of the functions $g^\alpha(\ \cdot\ )$ as the gap functions.*

Example 4 (The set $\mathcal{C}$ equal to $\mathbb{R}^+$) *To illustrate the last point on a very simple example, let us take the example an admissible set $\mathcal{C}$ equal to $\mathbb{R}^+$. The complementarity relation*

$$-dr \in N_{\mathcal{C}}(q) \Leftrightarrow 0 \leqslant q \perp dr \geqslant 0$$

is replaced by

$$-dr \in N_{T_{\mathcal{C}}(q)}(v^+) \Leftrightarrow \begin{cases} \textit{if } q \leqslant 0, \textit{ then } 0 \leqslant v^+ \perp dr \geqslant 0 \\ \\ \textit{if } q > 0, \textit{ then } dr = 0 \end{cases}$$

Second order Moreau's sweeping process Moreau's scheme [52, 54, 55] for scleronomous holonomic perfect unilateral constraints is based on a formulation of unilateral constraints in terms of local velocities together with the Newton impact law (see [51, 13, 66] for details).

Moreau [54] proposed a compact formulation of the impact law as an MDI,

$$-\mathrm{d}I \in N_{T_{\mathbb{R}_+^m}(g(q(t)))}(U^+(t) + \rho U^-(t))$$

where $T_{\mathbb{R}_+^m}(y)$ stands for the tangent cone to $\mathbb{R}_+^m$ at y [**?** 63]. Finally, we obtain an MDI, the so-called *Moreau sweeping process*,

$$M(q(t))\, \mathrm{d}v - F(t, q(t), v^+(t))\, \mathrm{d}t \in -G(q(t)) N_{T_{\mathbb{R}_+^m}(y(t))}(U^+(t) + \rho U^-(t)).$$

Note FP: pas de Moreau66-67 dans la biblio. Tu penses à quel papier?

[8] This definition permits to compute the tangent cone even when the constraints are violated, which is needed numerically.

Remark 6 *This formulation of the unilateral constraints together with Newton's impact law can be interpreted as an index reduction technique in DAE theory. If the constraints on the generalized coordinates are satisfied for the initial conditions, they are also satisfied at any time.*

Coulomb's friction Let us consider now Coulomb's friction. In such a case when more complex contact laws are considered, the pure Lagrangian modeling of constraints is not sufficient. Indeed, the use of the Jacobian matrix of the constraints $G^\top(t, q(t))$ in order to define the normal to the constraints is not necessarily convenient to introduce richer mechanical behaviors at the interface. Hence, we introduce for each contact α a local orthonormal frame at contact point C^α composed of a normal vector n^α and two tangent vectors t^α and s^α. In this frame, the local velocity at contact U^α and the reaction force λ^α are decomposed in its normal and tangent part as

$$\begin{aligned} U^\alpha &= U^\alpha_{\mathrm{N}} n^\alpha + U^\alpha_{\mathrm{T}}, \quad U^\alpha_{\mathrm{N}} \in I\!R, \quad U^\alpha_{\mathrm{T}} \in I\!R^2, \\ \lambda^\alpha &= \lambda^\alpha_{\mathrm{N}} n^\alpha + \lambda^\alpha_{\mathrm{T}}, \quad \lambda^\alpha_{\mathrm{N}} \in I\!R, \quad \lambda^\alpha_{\mathrm{T}} \in I\!R^2. \end{aligned}$$

Note that the operator $G(q)$ in (8) that links variables expressed in the local frame to generalized variables is not necessarily the gradient of some constraints.

Coulomb's friction is expressed in a disjunctive form as

$$\begin{cases} \text{if } U_{\mathrm{T}} = 0 & \text{then } \lambda \in \mathbf{C} \\ \text{if } U_{\mathrm{T}} \neq 0 & \text{then } ||\lambda_{\mathrm{T}}|| = \mu |\lambda_{\mathrm{N}}| \\ & \text{and there exists a scalar } a \geqslant 0 \text{ such that } \lambda_{\mathrm{T}} = -a U_{\mathrm{T}} \end{cases}$$

where $\mathbf{C} = \{\lambda, ||\lambda_{\mathrm{T}}|| \leqslant \mu |\lambda_{\mathrm{N}}| \}$ is the Coulomb friction cone. Let us introduce the modified velocity $\widehat{U}$ [25] defined by

$$\widehat{U} = U + \mu \, ||U_{\mathrm{T}}|| n.$$

With the Signorini condition at the velocity level, this notation provides us with a synthetic form of the Coulomb friction as

$$-\widehat{U} \in \mathrm{N}_{\mathbf{C}}(\lambda),$$

where $\mathrm{N}_{\mathbf{C}}$ is the normal cone to $\mathbf{C}$ [63], or equivalently,

$$\mathbf{C}^* \ni \widehat{U} \perp \lambda \in \mathbf{C},$$

where $\mathbf{C}^* = \{v \in I\!R^n \mid r^\top v \geqslant 0, \forall r \in \mathbf{C}\}$ is the dual cone of C. For more details on this formulation and its theoretical interest, we refer to [10]. The sliding is depicted in Figure 14

In this form, the numerical time integration of systems with Coulomb's friction is similar to case with only Signorini's condition written in terms of complementarity at the velocity level. The standard schemes and the new approaches developed in the sequel directly apply to the case with Coulomb's friction.

Special instances of nonsmooth mechanical systems Nonsmooth mechanical systems or NonSmooth MultiBody Systems (NSMBS) can be defined in a very general setting by the

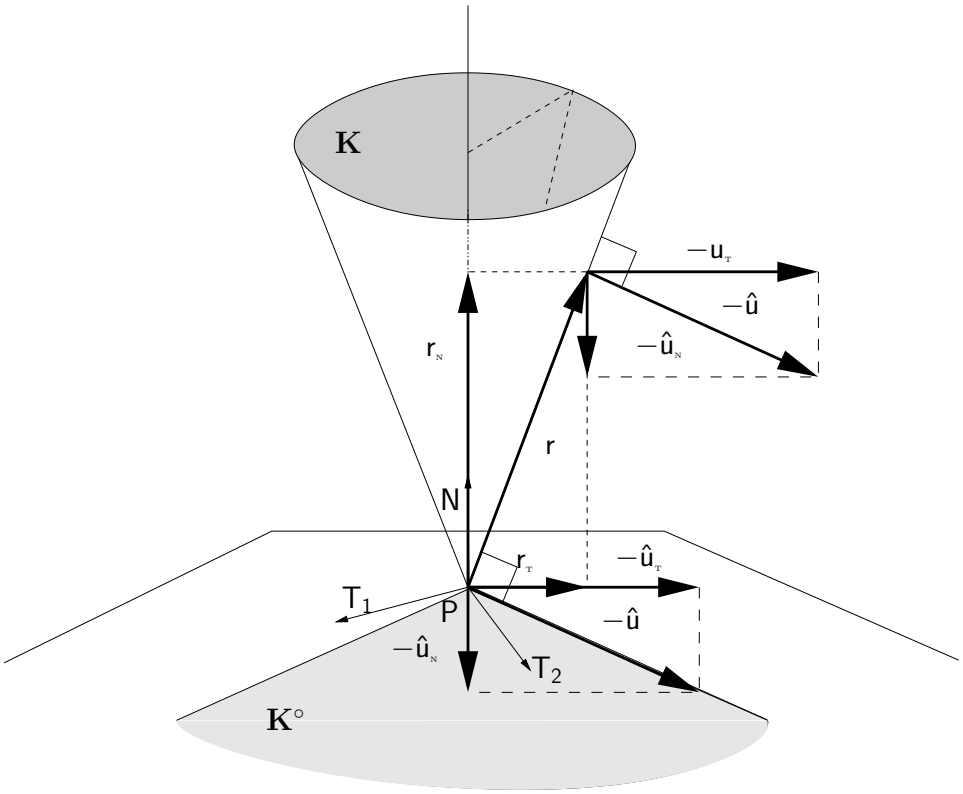


Figure 14: Coulomb's friction law in the sliding case.

following set of relations

$$\begin{cases} M(q)\dot{v} = F(t,q,v) + G(t,q)\lambda \\ \dot{q} = v \\ y = g(t,q,v) \\ 0 \in S(y,\lambda) + T(y,\lambda) \\ \mathcal{F}(v^+, v^-, q, t) \ni 0 \end{cases}$$

The inclusion (5d) defines the relation between the multiplier λ and the output y. As previously mentioned, the function $S\colon \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^m$ is assumed to be continuously differentiable and $T\colon \mathbb{R}^m \times \mathbb{R}^m \rightrightarrows \mathbb{R}^m$ is a multivalued mapping with a closed graph. Finally, if the evolution is nonsmooth, a reinitialization rule (or update rule), also known as an impact law, has to be added to specify the behavior of the velocities.

The most simple instance of multibody dynamical systems are mechanical systems subjected to nonsmooth bilateral constraints, the multibody systems subjected to perfect unilateral constraints

Definition 6 (Multibody systems with bilateral constraints (joints)) *Choosing $S(y,\lambda) = y = g(t,q)$ and $T(y,\lambda) = 0$, a multibody system subjected to nonsmooth bilateral is defined by the following Differential Algebraic Equation (DAE)*

$$\begin{cases} M(q)\dot{v} = F(t,q,v) + G(t,q)\lambda \\ \dot{q} = v \\ g(t,q) = 0 \end{cases}$$

If the constraints are not sufficiently smooth, let us say, g is only continuous in q, some nonsmooth solutions have to expected (see for more details Chap 11 in [35]) and then an impact law

$$0 \in \mathcal{F}(v^+, v^-, q, t)$$

has to be added.

Using inequalities, we get multibody systems subjected to perfect unilateral constraints.

Definition 7 (Multibody systems subjected to perfect unilateral constraints.) *Choosing $y = g(t,q)$, $S(y,\lambda) = \lambda$ and $T(y,\lambda) = \partial\psi_{\mathbb{R}_+}(y)$, where ψ_K is the indicator function of the set K and the symbol ∂ denotes the subdifferential in the sense of the Convex Analysis, we get*

$$\begin{cases} M(q)\dot{v} = F(t,q,v) + G(t,q)\lambda \\ \dot{q} = v \\ y = g(t,q) \\ 0 \leqslant y \perp \lambda \geqslant 0 \\ 0 \in \mathcal{F}(v^+, v^-, q, t) \text{ if } y = 0 \end{cases}$$

XXX complete the definition

Definition 8 (Multibody systems with scleronomous unilateral contact with Coulomb's friction)

Choosing $y = g(t,q)$, $S(y,\lambda) = \dots$ *and* $T(y,\lambda) = \dots$, *we get*

$$\begin{cases} M(q)\dot{v} = F(t,q,v) + r(t) \\ \dot{q} = v \\ y = g(q) \\ U(t) = G^{\top}(q)\, v(t) \\ r(t) = G(q)\lambda(t) \\ \widehat{U} = U + \mu\, ||U_{T}||n \\ \boldsymbol{C}^{*} \ni \widehat{U} \perp \lambda \in \boldsymbol{C} \\ 0 \in \mathcal{F}(v^{+}, v^{-}, q, t) \ \textit{if } y = 0 \end{cases}$$

XXX complete the definition

Definition 9 (Multibody systems with scleronomous unilateral contact with Coulomb's friction) *Choosing* $y = g(t,q)$, $S(y,\lambda) = \dots$ *and* $T(y,\lambda) = \dots$, *we get*

$$\begin{cases} M(q)\dot{v} = F(t,q,v) + r(t) \\ \dot{q} = v \\ y = g(q) \\ U(t) = G^{\top}(q)\, v(t) \\ r(t) = G(q)\lambda(t) \\ \widehat{U} = U + \mu\, ||U_{T}||n \\ \boldsymbol{C}^{*} \ni \widehat{U} \perp \lambda \in \boldsymbol{C} \\ 0 \in \mathcal{F}(v^{+}, v^{-}, q, t) \ \textit{if } y = 0 \end{cases}$$

More details on nonsmooth Lagrangian dynamical systems can be found in the following references [60, 18, 48, 5].

Link with Hybrid systems Once one side the nonsmooth mechanical system as in Definition 7 can be understood as an hybrid system with 2^m modes. A mode may be defined by the fact that each constraint $\alpha = 1 \dots m$, is active or not. In each mode, the solution is assumed to be smooth with discontinuities arising at transitions between two modes. On the other side, it can be considered as a single nonsmooth dynamical systems. Its solution is then defined as a global one, possibly nonsmooth, containing the instants of discontinuity. Usually, for Lagrangian systems, the coordinates are considered as absolutely continuous functions of time, the velocities as functions of bounded variations and accelerations as measures [64, 54, 51].

9 Other Classes of nonsmooth Dynamical systems.

Without entering into more details, the following system are usually considered to belong to the class of nonsmooth dynamical systems [18, 67, 5]:

- Ordinary Differential Equations (ODE) with only Lipschitz or absolutely continuous right-hand sides.

- Piecewise smooth systems
- Differential inclusions with compact and convex set-valued mappings (Filippov and Utkin's Theory.
- Differential normal cone inclusions.
- Measure differential inclusions.
- Evolution variational inequalities.
- Projected dynamical systems.

The Siconos platform aims at providing a scientific simulation software for all classes of systems for which some specific algorithms have been designed. Since Siconos is a project always under evolution, feedback form users drives the new development and implementation. For a comprehensive review of NSDS, we refer to the following monographs [18] and [67].

9.1 Comparison with the Hybrid Approach

For simple dynamics (constant or linear) and small systems (up to 100 degrees of freedom), the hybrid approach allows us to exploit the widespread analysis techniques for finite-state systems, such as the verification techniques to check some fundamental properties. For larger systems with fully nonlinear dynamics, it seems that these discrete techniques hardly apply.

In the case of NSDS, the mathematical properties of the solution and the associated numerical techniques allow the simulation and the analysis of large systems to be performed. We claim that the continuous approach is more efficient from the mathematical and the numerical point of view, rather than an event-driven or a discrete approach.

On the mathematical point of view, a NSDS can be considered as a unique time-continuous dynamical system which can encounter discontinuities and reinitializations. That leads to the definition of global solution of such systems in a class of appropriate functions or measures. The case of the constrained Lagrangian dynamics leads for example to the formulation of the dynamics in terms of measure differential inclusions [64, 54, 51] valid on the continuous part of the evolution as well as at the events. Such types of solutions and formulations can comprised complex sequences of events, like concurrent events or accumulation (Zeno), together with mathematical results such as global existence and uniqueness.

10 Simulation of NSDS

10.1 The NonSmooth Approach *vs.* the Hybrid Approach

From the numerical point of view, a nonsmooth approach of hybrid systems exploits the previous notion of solution defined everywhere in time. This fact leads to the design of powerful time–stepping schemes without explicit event-handling procedure. For the case of the Lagrangian dynamics, the measure differential inclusion is evaluated on a fixed time interval and the structural changes of the dynamics are taken into account in a weak sense. Contrary to event-driven schemes, such time-stepping ones are proved to be convergent even in the presence of events accumulations.

Another advantage of the nonsmooth approach of NSDS is the algebraic formulation of certain classes of state transitions at events. To shed more light on this aspect, the simple example of an ideal diode might be taken. This behavior can be modeled as a pure logical component thanks to an "if" statement as in Modelica [28]. Indeed recognizing that the curve of the graph in Figure **??** can be parametrized by a parameter s, we can defined the following Modelica script:

$$\begin{aligned} \mathsf{off} &= \mathsf{s} < 0 \\ \lambda &= \textbf{if} \quad \mathsf{off} \quad \textbf{then} \quad -\mathsf{s} \quad \textbf{else} \quad 0 \\ y &= \textbf{if} \quad \mathsf{off} \quad \textbf{then} \quad 0 \quad \textbf{else} \quad \mathsf{s} \end{aligned}$$

The same representations can be performed with ideal switches, piecewise linear model of MOS transistors. The main difficulties to view systems with ideal components this way is that for each new Boolean variable like off, two modes of the hybrid dynamical system are possible. If we introduce n-Boolean variables, in the worst case, 2^n modes have to be checked. Therefore the problem complexity is exponential.

On the contrary, in the nonsmooth approach the discretized problem at each step can be reformulated as a Linear Complementarity Problem (LCP) [24] of the form:

$$\begin{cases} w = Mz + q \\ 0 \leqslant w \perp z \geqslant 0 \end{cases}$$

Under some usual assumptions on the matrix M, (positiveness, n-step property), and on the vector q, numerical algorithms can be used with polynomial complexity, avoiding an exhaustive enumerative verification of each modes in an exponential time algorithm [59].

To conclude, from the mathematical point of view, the nonsmooth framework yields precise definitions of solutions together with uniqueness and existence results under appropriate assumptions. From the numerical point of view, the use of specific algorithms (time–stepping schemes, LCP solvers with polynomial complexity) leads to an efficient simulation environment. Therefore, the Siconos platform is based on these two features.

10.2 Event-detecting time–stepping schemes (Event–driven) and Event–capturing time–stepping schemes

Two types of methods are available in Siconos to numerically integrate in time NSDS.

- The first one is known as the *Event-driven method* where the time of discontinuities in the state or in its derivative, also called a nonsmooth event, is detected and located. This method can also known as the *nonsmooth event tracking method*. Between two events, the system is integrated with any standard ODE or Differential Algebraic Equation (DAE) integration routine, of suitable order according to the regularity of the system. This method for any standard ODE can be very efficient and of any order but suffers from several drawbacks. If the number of events is large, or worse infinite in a bounded time interval (Zeno), the time integration cannot efficiently advance in time. This is particularly the case when a finite accumulation of impacts is encountered. In this case, the simulation does not end in finite-time. Secondly, in practice, this method is very sensitive to numerical tolerances and accuracies used for the detection of events. Finally, such a method needs a reformulation of the generalized equation at different kinematic levels. This index-like

Method	Advantages / Weaknesses
Event tracking schemes	⊕ high accuracy integration of free flight motions
(a.k.a event-driven)	⊖ no proof of convergence
	⊖ sensibility to numerical thresholds
	⊖ reformulation of constraints at higher
Event capturing schemes	⊕ robust, stable and proof of convergence (Zeno)
(a.k.a time–stepping)	⊕ able to deal with finite accumulation
	⊕ low kinematic level for the constraints
	⊖ low order of accuracy even in free flight

Table 1: Qualitative comparisons of time–stepping schemes for nonsmooth dynamics

reduction contains some new conditional statements on the unilateral conditions, which leads to new numerical tolerances with theirs associated difficulties. Event-driven approaches are well-suited when the nonsmooth events are rare and well-separated in time.

- The other method is known as *time–stepping method* or *nonsmooth event capturing method*. In such a method, the time-integration is performed with a time step, which do not depend on the exact location of nonsmooth events. The advantages of this class of methods are the convergence proofs and the efficiency, even in the case of finite accumulation of impacts. It is also able to work without an accurate event detection. Finally, another practical interest of this method is that it needs lower kinematic reformulation, and even better no reformulation at all for the constraints. The major drawback of this method is its order: it is at best of first order for the impacts but also on the smooth solutions.

As we said the two possibilities are available on the Siconos software. There is also a possibility to take into account exogenous events in a time–stepping approach by controlling the integration through an event manager. The feature allows one to exploit the time–stepping abilities for the nonsmooth events and to drive the simulation with external events: control, switch, user input, ...

DETAILS SUR EVENT DRIVEN ET tIME STEPPING (ALGOs)

DO A COMPARISON WITH THE BOUNCING BALLL EXAMPLE

RECAP PROPERTIES ON A TABLEAU ==> DONE

11 Moreau Time-Stepping

Roughly speaking, the time-stepping method consists in the time-discretization of the whole system (dynamics + relations + non-smooth laws), leading to a so-called One-Step NonSmooth Problem (OSNSP) solved at each time step. The main stages of the process are:

1. integrate the dynamics without constraints, to get some “free” solutions.
2. construct and solve an OSNSP (a LCP for instance).
3. update the dynamics with the OSNSP solutions to get the full state update.

The figure below represents the architecture for classes related to `TimeStepping` simulation. TODO FIGURE

In the following sections, the systems are integrated over a time interval $[t_k, t_{k+1}]$ of length h. The approximation of any function $F(t, \dots)$ at the time t_k is denoted by F_k. Note that in the relations writing, we use upper case letters for all variables related to `DynamicalSystem` objects: $X, Q, \dots$ are concatenation of $x, q, \dots$ of the dynamical systems variables associated with the relation. We omit the parameter z most of the time.

We now present the integrator scheme for both First Order and Lagrangian systems.

11.1 First order systems

11.1.1 Time Discretization of the Dynamics

First Order Non Linear Systems With this most generic case (in SICONOS), the dynamics is

$$\begin{aligned} M\dot{x}(t) &= f(x,t,z) + r \\ x(t_0) &= x_0 \end{aligned}$$

with $r = r^d = \sum_\alpha r^\alpha, \alpha \in \mathcal{I}_d$, with $\mathcal{I}_d$ the set of all relations in which the current dynamical system, number d, is involved. In the following, the index d is also omitted.

The integration of the ODE over $[t_k, t_{k+1}]$ yields:

$$M \int_{t_k}^{t_{k+1}} \dot{x}\, \mathrm{d}t = \int_{t_k}^{t_{k+1}} f(t,x,z)\mathrm{d}t + \int_{t_k}^{t_{k+1}} r\, \mathrm{d}t.$$

The left-hand term is $M(x(t_{k+1}) - x(t_k)) \approx M(x_{k+1} - x_k)$. Right-hand terms are approximated with a θ-method:

$$\begin{aligned} \int_{t_k}^{t_{k+1}} f(t,x,z)\mathrm{d}t &\approx h\theta f(t_{k+1}, x_{k+1}, z) + h(1-\theta) f(t_k, x_k, z) \\ &\approx h\theta f_{k+1} + h(1-\theta) f_k, \end{aligned}$$

and the third integral is approximated with:

$$\int_{t_k}^{t_{k+1}} r\, dt \approx hr(t_{k+1}) \approx hr_{k+1}.$$

Then, we get the following "residue"

$$\begin{aligned} \mathcal{R}(x_{k+1}) &= M(x_{k+1} - x_k) - h\theta f_{k+1} - h(1-\theta) f_k - hr_{k+1} \\ &= \mathcal{R}^{free}(x_{k+1}) - hr_{k+1}. \end{aligned}$$

Note that we use the "free" notation for terms related to the smooth part of the system. We apply a Newton method to solve $\mathcal{R}(x_{k+1}) = 0$. The gradient of the residue according to x is

$$\nabla_x \mathcal{R}(x) = M - h\theta \nabla_x f(t,x),$$

and we get (index α corresponds to the Newton iteration number):

$$W_{k+1}^{\alpha}(x_{k+1}^{\alpha+1} - x_{k+1}^{\alpha}) = -\mathcal{R}(x_{k+1}^{\alpha}),$$

with

$$W_{k+1}^{\alpha} = M - h\theta \left[\nabla_x f\right](t_{k+1}, x_{k+1}^{\alpha}).$$

If we assume that W_{i+1}^{k} is invertible, we get the solution at the Newton iteration $\alpha + 1$ as

$$\begin{aligned} x_{k+1}^{\alpha+1} &= x_{k+1}^{\alpha} - (W_{k+1}^{\alpha})^{-1}\mathcal{R}^{free}(x_{k+1}^{\alpha}) + h(W_{k+1}^{\alpha})^{-1}r_{k+1}^{\alpha+1} \\ &= x_{k+1}^{free,\alpha} + h(W_{k+1}^{\alpha})^{-1}r_{k+1}^{\alpha+1}. \end{aligned}$$

First Order Linear Systems The dynamics of this class of system is:

$$\begin{aligned} M\dot{x}(t) &= A(t,z)x(t) + b(t) + r \\ x(t_0) &= x_0 \end{aligned}$$

For the integration of the ODE over a time step, we proceed as in the previous section for non-linear systems to get:

$$\mathcal{R}(x_{k+1}) = M(x_{k+1} - x_k) - h\theta(A_{k+1}x_{k+1} + b_{k+1}) - h(1-\theta)(A_k x_k + b_k) - hr_{k+1} = 0$$

or

$$(M - h\theta A_{k+1})x_{k+1} = (M + h(1-\theta)A_k)x_k + h\theta(b_{k+1} - b_k) + hb_k + hr_{k+1}.$$

We define $W_{k+1} = (M - h\theta A_{k+1})$ and assuming it is invertible, we get:

$$\begin{aligned} x_{k+1} &= W_{k+1}^{-1}\left[(M + h(1-\theta)A_k)x_k + h\theta(b_{k+1} - b_k) + hb_k\right] + hW_{k+1}^{-1}r_{k+1} \\ &= x_{k+1}^{free} + hW_{k+1}^{-1}r_{k+1}. \end{aligned}$$

First Order Linear Systems with time invariant coefficients

$$\begin{aligned} M\dot{x}(t) &= Ax(t) + b + r \\ x(t_0) &= x_0 \end{aligned}$$

Using the results of the previous section, the discretization is straightforward:

$$\begin{aligned} x_{k+1} &= x_k + hW^{-1}(Ax_k + b) + hW^{-1}r_{k+1} \\ &= x_k^{free} + hW^{-1}r_{k+1}, \end{aligned}$$

with a constant W:

$$W = (M - h\theta A).$$

11.2 Lagrangian systems

Lagrangian Non Linear Systems We provide in the following sections a time discretization method of the Lagrangian dynamical systems, consistent with the non smooth character of the solution.

$$\begin{aligned} M(q(t),z)\mathrm{d}v &= f_L(t,v^+(t),q(t),z)\mathrm{d}t + \mathrm{d}r \\ v^+(t) &= \dot{q}^+(t) \\ q(t_0) &= q_0 \\ \dot{q}(t_0^-) &= v_0 \end{aligned}$$

with

$$q(t) = q_0 + \int_{t_0}^{t} v^+(t)\mathrm{d}t.$$

Please note that $v^+(t)$ is the condensed form of $v(t^+)$, that is the right limit of v in t. The left-hand side is discretized while assuming that:

$$\int_{t_k}^{t_{k+1}} M(q(t),z)dv \approx M(q^*,z)(v_{k+1}-v_k)$$

As for first order non-linear systems, we use a θ-method to integrate the other terms, and obtain:

$$\int_{t_k}^{t_{k+1}} f_L(t,v^+(t),q(t),z)dt \approx h\theta f_L(t_{k+1},v_{k+1},q_{k+1},z) + h(1-\theta)f_L(t_k,v_k,q_k,z),$$

and for the last term, we set a new variable p_{k+1} such that:

$$\int_{t_k}^{t_{k+1}} dr \approx p_{k+1}.$$

Finally the full system discretization results in:

$$\begin{aligned} \mathcal{R}(v_{k+1},q_{k+1}) &= M(q*,z)(v_{k+1}-v_k) - h\theta f_{Lk+1} - h(1-\theta)f_{Lk} - p_{k+1} \\ &= \mathcal{R}^{free}(v_{k+1},q_{k+1}) - p_{k+1} \end{aligned}$$

The "free" notation still stands for terms related to the smooth part of the system. The displacement is integrated through the velocity with:

$$q_{k+1} \approx q_k + h\theta v_{k+1} + h(1-\theta)v_k$$

Injecting this into the residue leads to a function depending only on v_{k+1}, since state "k" is supposed to be known. A Newton method will be applied to solve $\mathcal{R}(v_{k+1}) = 0$, which requires the gradients of the residue. Assuming that the mass matrix evolves slowly with the configuration over time step, we get:

$$\nabla_{v_{k+1}}\left[M(q*,z)(v_{k+1}-v_k)\right] \approx M(q^*,z)$$

and denoting:

$$\begin{aligned} C_t(t,v,q) &= -\left[\frac{\partial f_L(t,v,q)}{\partial v}\right] \\ K_t(t,v,q) &= -\left[\frac{\partial f_L(t,v,q)}{\partial q}\right], \end{aligned}$$

we get (index α corresponds to the Newton iteration number):

$$W(t_{k+1}^{\alpha}, v_{k+1}^{\alpha}, q_{k+1}^{\alpha}) \cdot (v_{k+1}^{\alpha+1} - v_{k+1}^{\alpha}) = -\mathcal{R}(v_{k+1}^{\alpha}),$$

with

$$W(t, v, q) = M(q^*, z) + h\theta C_t(t, v, q) + h^2\theta^2 K_t(t, v, q).$$

As an approximation for q^*, we choose:

$$\begin{aligned} q^* &\approx (1-\gamma)q_k + \gamma q_{k+1}^{\alpha} \\ &\approx q_k + h\gamma\left[(1-\theta)v_k + \theta v_{k+1}^{\alpha}\right], \end{aligned}$$

with $\gamma \in [0, 1]$. Moreover, if M is evaluated at the first step of the Newton iteration, with $v_{k+1}^{0} = v_k$, we get:

$$M(q^*) \approx M(q_k + h\gamma v_k).$$

Finally, if W is invertible, the solution at iteration $\alpha + 1$ is given by:

$$\begin{aligned} v_{k+1}^{\alpha+1} &= v_{k+1}^{\alpha} - (W_{k+1}^{\alpha})^{-1}\mathcal{R}^{free}(v_{k+1}^{\alpha}) + (W_{k+1}^{\alpha})^{-1}p_{k+1}^{\alpha+1} \\ &= v_{k+1}^{free,\alpha} + (W_{k+1}^{\alpha})^{-1}p_{k+1}^{\alpha+1}. \end{aligned}$$

Lagrangian Linear Systems with Time Invariant coefficients

$$\begin{aligned} Mdv + Cv^+(t) + Kq(t) &= F_{ext}(t, z) + p \\ q(t_0) &= q0 \\ \dot{q}(t_0^-) &= v_0 \end{aligned}$$

Proceeding as previously, with M constant and with the relation

$$f_L(t, v^+(t), q(t), z) = F_{ext}(t) - Cv^+(t) - Kq(t),$$

the integration is straightforward:

$$\mathcal{R}(v_{k+1}, q_{k+1}) = M(v_{k+1} - v_k) - h\theta\left[F_{ext}(t_{k+1}) - Cv_{k+1} - Kq_{k+1}\right] - h(1-\theta)\left[F_{ext}(t_k) - Cv_k - Kq_k\right] - p_{k+1}.$$

Using the displacement integration through the velocity,

$$q_{k+1} = q_k + h\left[\theta v_{k+1} + (1-\theta)v_k\right]$$

the solution of $\mathcal{R}(v_{k+1}, q_{k+1}) = 0$ is

$$W(v_{k+1} - v_k) = (-hC - h^2\theta K)v_k - hKq_k + h\left[\theta F_{ext}(t_{k+1}) + (1-\theta)F_{ext}(t_k)\right] + p_{k+1},$$

with W a constant matrix:

$$W = \left[M + h\theta C + h^2\theta^2 K\right],$$

and if W is invertible,

$$\begin{aligned} v_{k+1} &= v_k + W^{-1}\left[(-hC - h^2\theta K)v_k - hKq_k + h\theta F_{ext}(t_{k+1}) + h(1-\theta)F_{ext}(t_k)\right] + W^{-1}p_{k+1} \\ &= v_k^{free} + W^{-1}p_{k+1}. \end{aligned}$$

The free velocity v^{free} corresponds to the velocity of the system without any constraints.

11.2.1 Time discretization of the nonsmooth laws

A natural way to discretize the unilateral constraint leads to the following implicit discretization :

$$0 \leqslant y_{k+1} \perp \lambda_{k+1} \geqslant 0.$$

In the Moreau's time–stepping, we use a reformulation of the unilateral constraints in terms of velocity:

$$\text{If } y(t) = 0, \quad \text{then} \quad 0 \leqslant \dot{y} \perp \lambda \geqslant 0,$$

which leads to the following discretization:

$$\text{If} \quad y^p \leqslant 0, \quad \text{then} \quad 0 \leqslant \dot{y}_{k+1} \perp \lambda_{k+1} \geqslant 0,$$

where y^p is a prediction of the position at time t_{k+1}, for instance, $y^p = y_k + \frac{h}{2}\dot{y}_k$. If we want to introduce now the Newton impact law, we consider an equivalent velocity defined by

$$\dot{y}^e_{k+1} = \dot{y}_{k+1} + e\dot{y}_k$$

and we apply the constraints directly on this velocity:

$$\text{If} \quad y^p \leqslant 0, \quad \text{then} \quad 0 \leqslant \dot{y}^e_{k+1} \perp \lambda_{k+1} \geqslant 0.$$

Part III
Siconos Software

Siconos software is mostly written in C++ but also provides a Python interface. In both cases, Object-Oriented paradigm is largely used meaning that a proper understanding of Siconos structure and of its collection of objects is required to build and simulate a nonsmooth problem in a suited way. Even though a quick preview of these objects was proposed through the basic examples exposed in Part I, this part is dedicated to the actual implementation of NSDS and of their simulation inside Siconos Software. Main ideas and concepts used to modelize, simulate and control a NSDS are presented in the first section. Afterwards, an exhaustive and detailed review of each of these aspects and their related objects is proposed in Section 13 and 14..

12 General Principles of Modeling and Simulation

As explained in Part I, the central object of any Siconos simulation is the `Model`, which is roughly speaking the description of a NSDS and some instructions on how to simulate it during a given time period. So, the compulsory process to handle a problem with Siconos is first to build a `NonSmoothDynamicalSystem` (section 12.1) and then to describe a `Simulation` strategy (section 12.2). Additionally, a control input can be defined (section 12.3).
Note that the strict partition between modeling and simulation is a strong design choice and the way the software is written relies on this distinction.

12.1 NSDS Modeling in Siconos Software

Hopefully, Part II should have made the notion of Nonsmooth Dynamical System clearer. But now, the question is how to represent this NSDS from the software point of view? A NSDS can be viewed as a set of dynamical systems that may interact in a nonsmooth way. The modeling approach in Siconos platform consists in considering this NSDS as a graph with dynamical systems as nodes and nonsmooth "interactions" as edges, as shown on Figure 15(a). Consequently `DynamicalSystem` and `Interaction` are the fundamental objects of any Siconos modeling.

A `DynamicalSystem` object is no more than a set of ordinary differential equations that describes the dynamics of the state x of a single dynamical system, with some specific operators, initial conditions, the input due to the nonsmooth law (usually denoted r) and so on.

An `Interaction` object describes the way one or more dynamical systems are linked or may interact. It connects the "global" variables (i.e. the states of the concerned dynamical systems) and some so-called "local" variables, y and λ. $y = h(x)$ is denoted as the "output" from the `DynamicalSystem` while λ, such that $r = g(\lambda)$, is the "input" to the `DynamicalSystem`. An `Interaction` is always composed of:

- a `NonSmoothLaw` that describes the mapping between y and λ,
- a `Relation` object that describes the equations between the local variables (y, λ) and the

"global" ones from the `DynamicalSystem`(s).

For instance, if you consider a set of rigid bodies, the set of `Interaction` objects defines and describes what happens at contact. `Relation` will mostly describes the distances between bodies while `NonSmoothLaw` will express the constraints to be enforced at contact.
As an illustration, see Figure 15(b) which presents a simple case[9]: a single dynamical system with a single non smooth interaction (i.e. constraint). A complete review of dynamical systems available in Siconos is given in Section 13.1. In the same way, all the various possibilities for `Relation` and `NonSmoothLaw` objects are detailed in Sections 13.2 and 13.3.
As summarized on the UML graph on Figure 24, building a problem in Siconos relies on the proper identification and construction of some `DynamicalSystem`s and of all the potential `Interaction`s, gathered in a `NonSmoothDynamicalSystem`. The next question, tackled in the following section, is how to simulate this problem.

12.2 Simulation Strategies for the NSDS Behavior

Once a NSDS has been defined, it is necessary to build a `Simulation` object, to define the way the NSDS will be integrated through the time period defined in the `Model`. More precisely, the way the dynamics is integrated and the computation of the nonsmooth input remains to be specified.

First of all, let us introduce the `Event` object, which is characterized by a type and a time of occurrence. Each event has a `process` method which defines a list of actions that are executed when this event occurs. These actions obviously depend on the type of the `Event`.
For example, if an event-driven strategy is chosen (see 10.2), some nonsmooth time events, namely `NonSmoothEvent`, are defined and imply specific actions at the time of these events. (BE MORE PRECISE?)

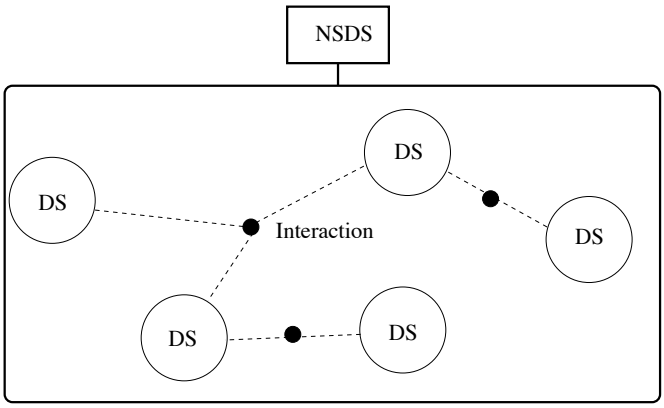
WOULDN'T IT BE BETTER TO INTRODUCE SENSOR AND ACTUATOR EVENTS IN THE CONTROL PART, TO SIMPLIFY THIS INTRO AND AVOID SOME SORT OF CONFUSION? IF SO, COMMENT THE 3 LINES BELOW For the `SensorsEvent` and `ActuatorEvent` related to control tools (see Section 12.3), an action is performed for both time–stepping and event–driven strategy at the times defined by the Control law. Finally, thanks to a registration mechanism, user–defined events can be added.

The set of events is handled by an `EventsManager` which belongs to the `Simulation` and will lead the process: systems integration is always done between a "current" and a "next" event. Moreover, during simulation, `Event`s of different types may be added or removed, for example when the user creates a `Sensor` or when an impact is detected.

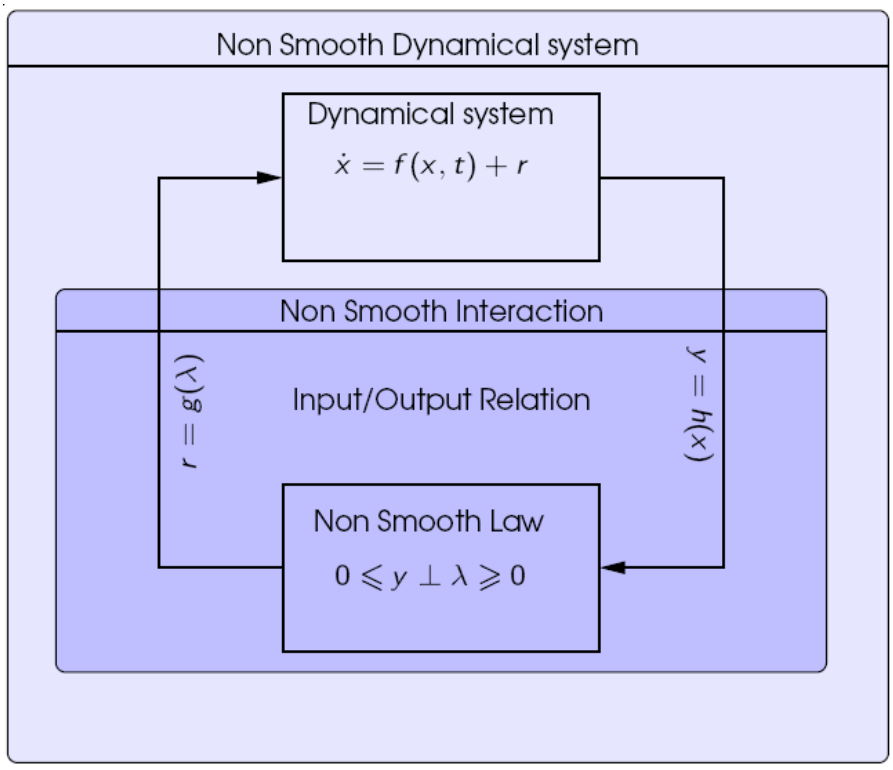
This notion of `Event` being explained, let us turn our attention to the proper construction of the `Simulation`. First of all, it is necessary to define a discretization of the time range (remind that initial and final time are part of the `Model`) using a `TimeDiscretisation` object, to set the number of time steps and their respective sizes. The time instants of this discretization define specific `Event`s, `TimeDiscretisationEvent` objects used to initialize the `EventsManager`. Thereafter, to complete the `Simulation`, we need:

- some instructions on how to integrate the smooth dynamics over a time-step, that is the role of the `OneStepIntegrator` objects (Section 14.1).

[9] Bouncing Ball and Diode Bridge examples from Part I both fall within this framework.



(a) The graph structure of a complex NSDS with `DynamicalSystem` objects as nodes and nonsmooth `Interaction` object as branches



(b) A simple `NonSmoothDynamicalSystem` with one `DynamicalSystem` object and one `Interaction`

Figure 15: Siconos NonSmooth Dynamical System Modeling Principle.

- some details on how to formalize and solve the nonsmooth problems when they occur, this is done with the `OneStepNSProblem` objects (Section 14.2).

To summarize, given a `Model` handling a `NonSmoothDynamicalSystem`, its numerical simulation is done through a `Simulation` object, composed of a `TimeDiscretisation`, a set of `OneStepIntegrator` plus a set of `OneStepNSProblem`.

The whole simulation process is led by the chosen type of strategy, either time–stepping or event–driven (see Section 10.2). To proceed, one need to instantiate one of the classes that inherits from `Simulation`: `TimeStepping` or `EventDriven`. TALK ABOUT D1MINUS?

12.3 Control Tools TODO Olivier H.

In Siconos, some control can be applied on a NSDS. The principle is to get information from the systems thanks to some `Sensor` objects, used by some `Actuator` objects to act on the NSDS components. Each `Sensor` or `Actuator` object has its own `TimeDiscretisation` object, a list of time instants where data are to be captured for sensors or where action occurs for actuators. Those instants are scheduled as events into the simulation's `EventsManager` object and thus processed when necessary.
The whole control process is handled thanks to a `ControlManager` object, which is composed of a set of `Sensor` objects and another set of `Actuator` objects. The `ControlManager` object "knows" the `Model` object and thus all its components.

Each `DynamicalSystem` object has a specific variable, named z, which is a vector of discrete parameters (see section 13.1). To control the systems with a sampled control law, the `Actuator` object sets the values of z components according to the user instructions.

13 NSDS modeling

In the following paragraphs, we turn our attention to the specific types of systems, relations and laws available in the platform.

Shall we write something (where?) about plugin mechanism, used to define all function-like operators below?

13.1 Dynamical Systems

I remove the generic interface description which is, I think , useless: -> not present in any example (and indeed probably unusable ...) To be discussed. `DynamicalSystem`s objects represent sets of ordinary differential equations, describing the dynamics of systems. They are sorted into three categories, First Order, Lagrangian (second order) and Newton-Euler systems and then specialized according to the type of their operators (linear, time-invariant, ...), as illustrated on Figure 16. Notice that all classes inherit from the `DynamicalSystem` class.

In the following sections, we denote:

- n the dimension of the system
- t the time,

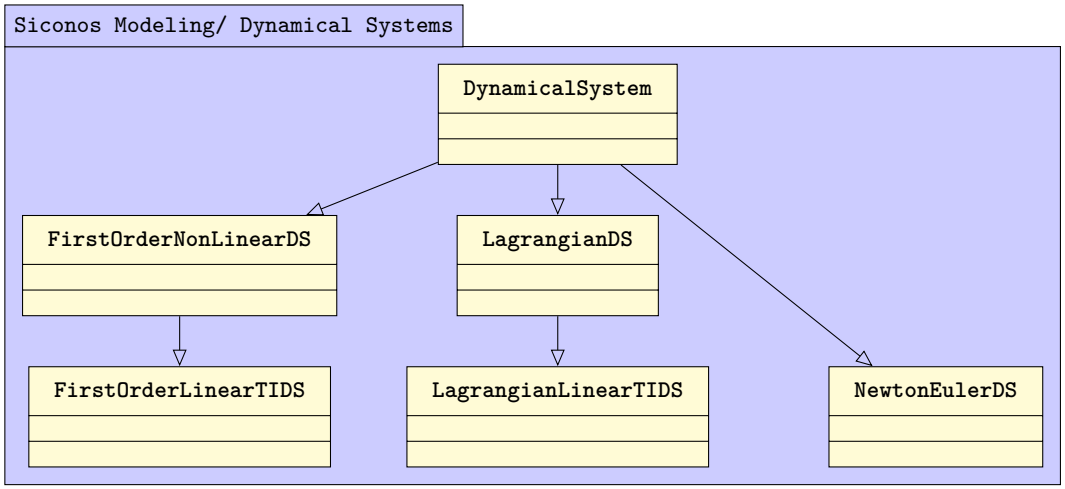


Figure 16: `DynamicalSystem` inheritance class diagram (non exhaustive)

- $x \in I\!R^n$ the state [10]
- $z \in I\!R^s$ a set of discrete states, set by user, that may be used to set some control or perturbation parameters.

13.1.1 First Order Dynamical Systems

`FirstOrderNonLinearDS`. This class describes the nonlinear first order dynamical systems as,

$$M\dot{x}(t) = f(t, x(t), z) + r,$$

with M a $n \times n$ matrix, $f(x, t, z)$ the smooth vector field and r the input due to the nonsmooth behavior. By default, the matrix M is considered as the identity matrix. $\nabla_x f(x, t, z)$ must be explicitly provided.

`FirstOrderLinearDS`. This class describes the first order linear dynamical systems as

$$M\dot{x}(t) = A(t, z)x(t) + b(t, z) + r. \tag{6}$$

`FirstOrderLinearTIDS`. This class describes the first order linear time invariant dynamical systems as

$$M\dot{x}(t) = Ax(t) + b + r.$$

Electrical circuits with linear smooth components and ideal ones fit this formalism, as shown in the Diode Bridge example in Section **??**. More examples can be found in [2, 3]. In the same way, these classes are used to model the smooth part of relay systems and sliding mode controlled systems [7, 9, 40, 41] (Filippov's differential inclusions) and of piecewise–smooth systems [32, 11].

[10] The typical dimension of the state vector can range between a few degrees of freedom and more than several hundred thousand, for example in mechanical or electrical systems. The implementation of the software has been done such that it is possible to deal either with small or large problems.

13.1.2 Lagrangian Dynamical Systems

Lagrangian are second order systems and the generalized coordinates are denoted $q \in R^{ndof}$ such that $n = 2ndof$ and

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

`LagrangianDS.` . Lagrangian nonlinear dynamical systems in the form,

$$M(q,z)\ddot{q} + F_{\text{gyr}}(\dot{q},q,z) + F_{\text{int}}(t,\dot{q},q,z) = F_{\text{ext}}(t,z) + p, \tag{7}$$

where F_{gyr} denotes the gyroscopic forces, F_{int} the internal forces and F_{ext}, the external forces, depending only on time. This formalism corresponds to Mechanics, and can be written in a simpler manner as:

$$M(q,z)\ddot{q} = F(t,\dot{q},q,z) + p$$

The full-form (7) with several operator has been designed to fit with different users habits, depending on the application field (Multibody mechanics, Robotics, Solid and structures Mechanics through Finite Element Method (FEM)).

Obviously, all gradients of the operator are required to define properly the systems.

`LagrangianLinearTIDS.` Lagrangian linear and time invariant coefficients systems:

$$M\ddot{q} + C\dot{q} + Kq = F_{\text{ext}}(t,z) + p$$

where C and K are respectively the classical viscosity and stiffness matrices.

13.1.3 Newton–Euler Dynamical System

`NewtonEulerDS.` Newton Euler dynamical systems for which the equation of motion is usually stated as :

$$\left\{ \begin{array}{rcl} M\dot{v}_g & = & F_{ext} \\ I\dot{\omega} + \omega \times I\omega & = & M_{ext} \end{array} \right.$$

where

- for $m \in I\!R$, $M = \text{diag}(m,m,m)$ is a scalar matrix
- $I \in I\!R^{3\times 3}$ is the inertia matrix relative to the center of mass
- $\dot{v}_g \in I\!R^3$ is the acceleration vector of the center of mass
- $\omega \in I\!R^3$ is the angular velocity vector expressed in the frame attached to the body
- $F_{ext} \in I\!R^3$ and $M_{ext} \in I\!R^3$ are the external applied forces and torques

In order to extend the equations of motion (3) to the Newton–Euler formalism, the vector of parameters q contains the position of the center of mass x_g and the four components of a quaternion q_o.

$$q = \begin{pmatrix} x_g \\ q_o \end{pmatrix} \in I\!R^7,$$

and the $T(q_0)$ operator (4) is

$$\begin{aligned} T(q_o)\colon I\!R^6 &\to I\!R^7 \\ \begin{pmatrix} \dot{x}_g \\ \omega \end{pmatrix} &\mapsto \begin{pmatrix} \dot{x}_g \\ \dot{q}_o \end{pmatrix} \end{aligned}$$

with

$$\dot{q}_o = \frac{1}{2}(0, \omega) q_o,$$

where $(0, \omega)$ is the quaternion with 0 as scalar part and ω, the angular velocity, as vector part.

13.2 Relations

As shown in Part I and **??**, some local variables must be defined to describe the behavior for each constraint. Indeed, a mapping is required to connect these local variables, denoted y and λ, and the global variables of the dynamical systems (either the pair (x, r) or (q, p) depending on the system type). This is the role of the `Relation` class and its heirs. They define these mappings through some general algebraic equations:

$$\begin{aligned} y &= h(t, \bar{\mathsf{x}}, \lambda, \bar{\mathsf{z}}) \\ \bar{\mathsf{r}} &= g(t, \lambda, \bar{\mathsf{x}}, \bar{\mathsf{z}}). \end{aligned}$$

"Topped" letters like $\bar{\mathsf{x}}, \bar{\mathsf{z}}$ and $\bar{\mathsf{r}}$ are used to represent `BlockVectors` defined as the concatenation of some state vectors of `DynamicalSystems`. Indeed an `Interaction` can concern 1 or 2 `DynamicalSystems` ds1 and ds2, like when 2 rigid bodies collide. In this case, we have the following definition:

$$\bar{\mathsf{x}} \coloneqq \begin{bmatrix} x_{ds1} \\ x_{ds2} \end{bmatrix} \quad \bar{\mathsf{z}} \coloneqq \begin{bmatrix} z_{ds1} \\ z_{ds2} \end{bmatrix} \quad \bar{\mathsf{r}} \coloneqq \begin{bmatrix} r_{ds1} \\ r_{ds2} \end{bmatrix}.$$

If there is only one `DynamicalSystem` concerned by the `Interaction`, we have

$$\bar{\mathsf{x}} \coloneqq x_{ds} \qquad \bar{\mathsf{z}} \coloneqq z_{ds} \qquad \bar{\mathsf{r}} \coloneqq r_{ds}.$$

As an example, consider the bouncing ball and the case where two moving bodies may be in contact. In the first situation, the local variable y is the distance to the ground and depends only on the position of the ball, meaning that $y = h(t, \bar{\mathsf{x}} = x_{ball}, z_{ball})$. When two bodies interact, the local variable y is still the distance between bodies but now depends on both positions, $y = h(t, \bar{\mathsf{x}} = \begin{bmatrix} x_{ds1} \\ x_{ds2} \end{bmatrix}, ...)$.

The `Relation` class organization mirrors the one for the `DynamicalSystem`s as shown on Figure 17. There are 3 types of `Relation`, corresponding to each type of `DynamicalSystem`: `FirstOrderR`, `LagrangianR` and `NewtonEulerR`. Each type is purely virtual and derived to build usable relations, defined by the properties of the mapping h, g. INCLUDE GRAPH AS FOR DS ==> Done. Is

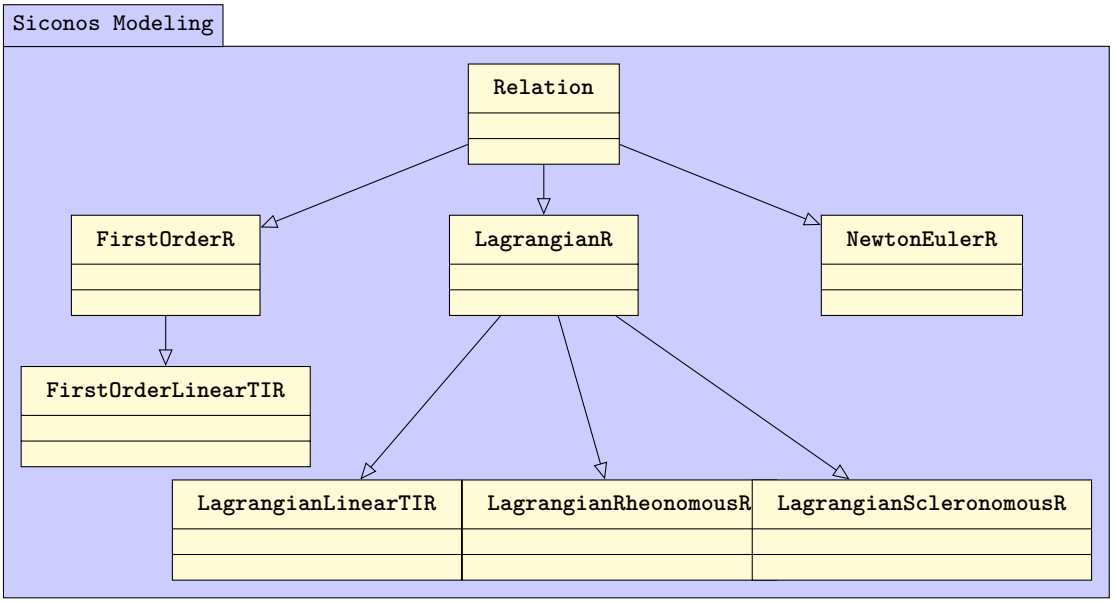


Figure 17: `Relation` inheritance class diagram

it sufficient ?

SPECIFY IN WHICH EXAMPLE EACH RELATION IS USED

TALK ABOUT THE NEWTON LOOP HERE

13.2.1 First Order Relation

`FirstOrderNonLinearR`. These are the most generic first order nonlinear relations:

$$\begin{aligned} y &= h(t, \bar{x}, \lambda, \bar{z}) \\ \bar{r} &= g(t, \lambda, \bar{x}, \bar{z}) \end{aligned}$$

`FirstOrderLinearTIR`. First order linear and time invariant relations:

$$\begin{aligned} y &= C\bar{x} + F\bar{z} + D\lambda + e \\ \bar{r} &= B\lambda \end{aligned} \tag{8}$$

Once again, see for instance the Diode Bridge example in Part I.

13.2.2 Lagrangian Relation

`LagrangianScleronoumousR.` Scleronomic constraints case, where the relation depends only on the global coordinates of the Dynamical Systems.

$$\begin{aligned} y &= h(\bar{\mathsf{q}}, \bar{\mathsf{z}}) \\ \dot{y} &= G_0(\bar{\mathsf{q}}, \bar{\mathsf{z}})\dot{\bar{q}} \\ \bar{\mathsf{p}} &= \nabla^T h(\bar{\mathsf{q}}, \bar{\mathsf{z}})\lambda = G_0^T(\bar{\mathsf{q}}, \bar{\mathsf{z}})\lambda \end{aligned}$$

with

$$G_0(\bar{\mathsf{q}}, \bar{\mathsf{z}}) = \nabla_{\bar{\mathsf{q}}} h(\bar{\mathsf{q}}, \bar{\mathsf{z}})$$

`LagrangianRheonomousR.` Add a time dependence :

$$\begin{aligned} y &= h(\bar{\mathsf{q}}, t, \bar{\mathsf{z}}) \\ \dot{y} &= G_0(\bar{\mathsf{q}}, t, \bar{\mathsf{z}})\dot{\bar{q}} + \frac{\partial h}{\partial t}(\bar{\mathsf{q}}, t, \bar{\mathsf{z}}) \\ \bar{\mathsf{p}} &= G_0^T(\bar{\mathsf{q}}, t, \bar{\mathsf{z}})\lambda \end{aligned}$$

with

$$G_0(\bar{\mathsf{q}}, t, \bar{\mathsf{z}}) = \nabla_{\bar{\mathsf{q}}} h(\bar{\mathsf{q}}, t, \bar{\mathsf{z}})$$

`LagrangianCompliantR.` There, the relation depends on λ. For instance in the mechanical case, this may correspond to a spring, since it links more or less a force to a displacement.

$$\begin{aligned} y &= h(\bar{\mathsf{q}}, \lambda_0, \bar{\mathsf{z}}) \\ \dot{y} &= G_0(\bar{\mathsf{q}}, \lambda_0, \bar{\mathsf{z}})\dot{\bar{q}} + G_1((\bar{\mathsf{q}}, \lambda_0, \bar{\mathsf{z}})\lambda_1 \\ \bar{\mathsf{p}} &= G_0^T(\bar{\mathsf{q}}, \lambda_0, \bar{\mathsf{z}})\lambda_0 \end{aligned}$$

with

$$\begin{aligned} G_0(\bar{\mathsf{q}}, \lambda_0, \bar{\mathsf{z}}) &= \nabla_{\bar{\mathsf{q}}} h(\bar{\mathsf{q}}, \lambda_0, \bar{\mathsf{z}}) \\ G_1(\bar{\mathsf{q}}, \lambda_0, \bar{\mathsf{z}}) &= \nabla_{\lambda_0} h(\bar{\mathsf{q}}, \lambda_0, \bar{\mathsf{z}}) \end{aligned}$$

and λ_0 the multiplier corresponding to y, while λ_1 corresponds to $\dot{y}$.

`LagrangianLinearR.` Linear and time invariant relations between local and global variables.

$$\begin{aligned} y &= H\bar{\mathsf{q}} + D\lambda + F\bar{\mathsf{z}} + b \\ \bar{\mathsf{p}} &= H^T\lambda \end{aligned}$$

C is used in place of H in the soft. But H is more classical in Mechanics. Which one must we choose ...?

13.2.3 Newton–Euler Relations

For Newton–Euler systems, unilateral and bilateral constraints are implemented for the time invariant case only.

The Newton–Euler relation, with the help of the T operator (4) may be stated as:

$$y = h(\bar{\mathsf{q}})$$
$$\dot{y} = \nabla_{\bar{\mathsf{q}}} h(\bar{\mathsf{q}})\, T(q_o) \begin{pmatrix} \dot{x}_g \\ \omega_. \end{pmatrix}$$

This is how the unilateral constraint is involved in the implementations of contacts without friction in the class `NewtonEulerFrom1DLocalFrameR` and of contacts with friction in the class `NewtonEulerFrom3DLocalFrameR`.

Bilateral relations for one or two bodies are implemented in `PivotJointR`, `PrismaticJointR`, `KneeJointR`.

13.3 Nonsmooth Laws

Nonsmooth laws are required to characterize the behavior of the local variables of the constraints. `NonSmoothLaw` object is a required part of the `Interaction` class, with the above `Relations`. Different laws are available in Siconos as shown on Figure 18. WHERE IS THIS PiecewiseLinearNSL implemented ?

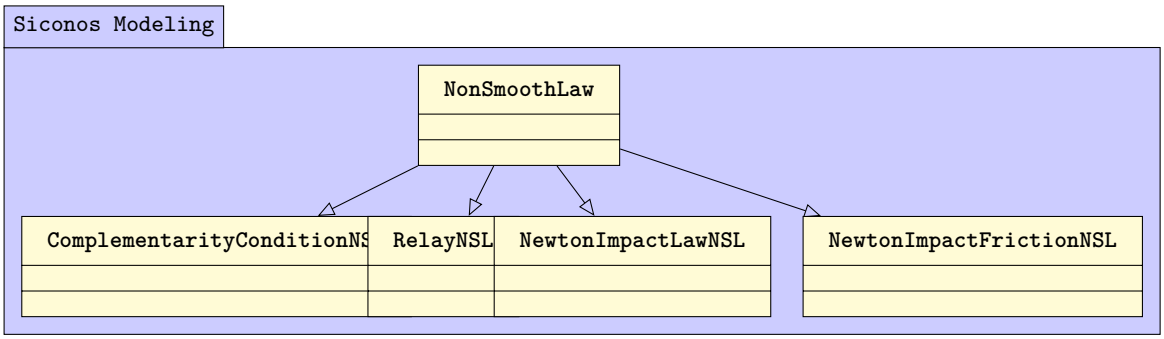


Figure 18: `NonSmoothLaw` inheritance class diagram

`ComplementarityConditionNSL`. A class which models the classical complementarity condition:

$$0 \leqslant y \perp \lambda \geqslant 0$$

`NewtonImpactNSL`. A class which models the unilateral contact with the Newton's impact law, known also as the Moreau's impacting rule:

$$\text{if } y(t) = 0, \quad 0 \leqslant \dot{y}(t^+) + e\dot{y}(t^-) \perp \lambda \geqslant 0$$

`RelayNSL`. A simple relay mapping as

$$\lambda \in -\text{sign}(y)$$

with a multivalued sign function: $\text{sign}(0) = [-1, 1]$.

`NewtonImpactFrictionNSL`. Unilateral contact with Coulomb's friction in 2D and 3D as

$$y = [y_{\boldsymbol{n}}, y_{\boldsymbol{t}}]^T, \lambda = [\lambda_{\boldsymbol{n}}, \lambda_{\boldsymbol{t}}]^T$$

$$\text{if} \quad y_{\boldsymbol{n}} = 0, \begin{cases} 0 \leqslant \dot{y}_{\boldsymbol{n}} \perp \lambda_{\boldsymbol{n}} \geqslant 0 \\ \dot{y}_{\boldsymbol{t}} = 0, \|\lambda_{\boldsymbol{t}}\| \leqslant \mu \lambda_{\boldsymbol{n}} \\ \dot{y}_{\boldsymbol{t}} \neq 0, \lambda_{\boldsymbol{t}} = -\mu \lambda_{\boldsymbol{n}} \text{sign}(\dot{y}_{\boldsymbol{t}}) \end{cases}$$

`PiecewiseLinearNSL`. One-dimensional piecewise linear set-valued mapping with fill in graphs as depicted on the Figure 19

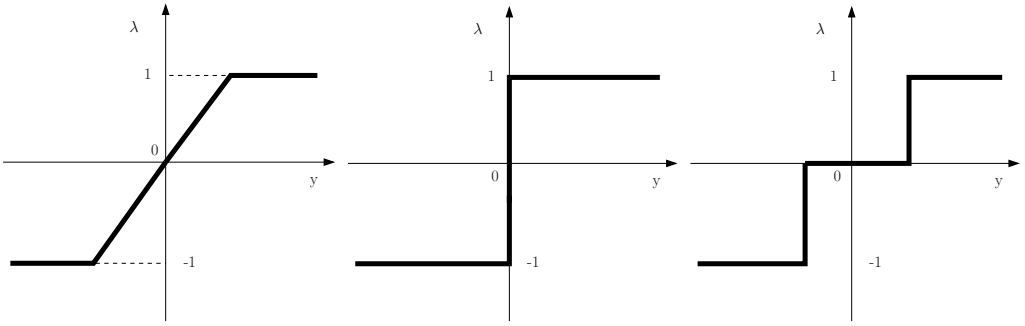


Figure 19: Some multivalued piecewise linear laws: saturation, relay, relay with dead zone

`MultipleImpactNSL`.

`MixedComplementarityConditionNSL`. TODO : document the two laws above. Somebody knows them well? The doc in MultipleImpactNSL is, well, quite light ... Is there a paper about it somewhere?

14 Simulation Related Components

14.1 Integration of the Dynamics

To integrate the dynamics over a time-step or between two events, `OneStepIntegrator` objects have to be defined. Two types of integrators are available at the time in the platform, listed below and represented on Figure 20.a:

- `Moreau` class for Moreau's time stepping scheme, based on a θ-method.
- `Lsodar` class for the Event Driven strategy; this class is an interface for LSODAR, odepack integrator (see `http://www.netlib.org/alliant/ode/doc`).

14.2 Formalization and Solving of the Nonsmooth Problems

Depending on the encountered situation, various formalizations for the nonsmooth problem are available.

- `LCP` class which describes the Linear Complementarity Problem,

$$\Big\{ w = Mz + q0 \leqslant w \perp z \geqslant 0$$

- `FrictionContact2D(3D)` class, for two(three)-dimensional contact and friction problems
- `QP` class for the Quadratic Programming problem

$$\begin{cases} \min \frac{1}{2} z^T Q z + z^T p \\ z \geqslant 0 \end{cases}$$

- `Relay` class for the relay problem

From a practical point of view, the solving of nonsmooth problems relies on low level algorithms (from the Siconos/Numerics package). The available nonsmooth solvers for LCP are LexicoLemke, PGS (Projected Gauss Seidel), QP (Quadratic Programming), CPG (Conjugate Projected Gradient).

Figure 20: (a) One-Step Integrators classes - (b) One-Step Nonsmooth Problem classes.

Part IV
Siconos Software Design

Objectives of this part : clarify Siconos structure (which component is used for what) and expose the tools/patterns used to improve the soft

- a description of the components of Siconos
- the specificities of Siconos implementation (shared pointers, boost graph, ublas, xml in python ... anything that improve the functionalities, perf and so on)

15 Overview

Siconos is composed of three main parts: Numerics, Kernel and Front-End, as represented on Figure 21 below.

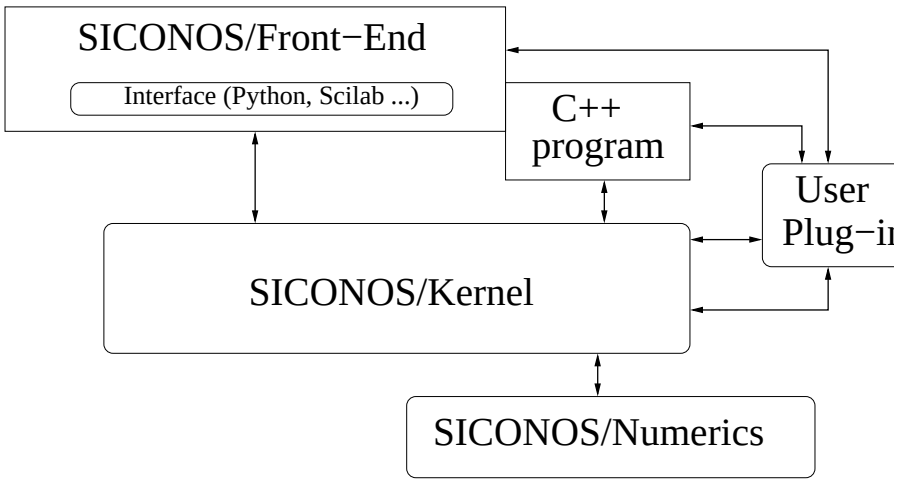


Figure 21: General design of Siconos software

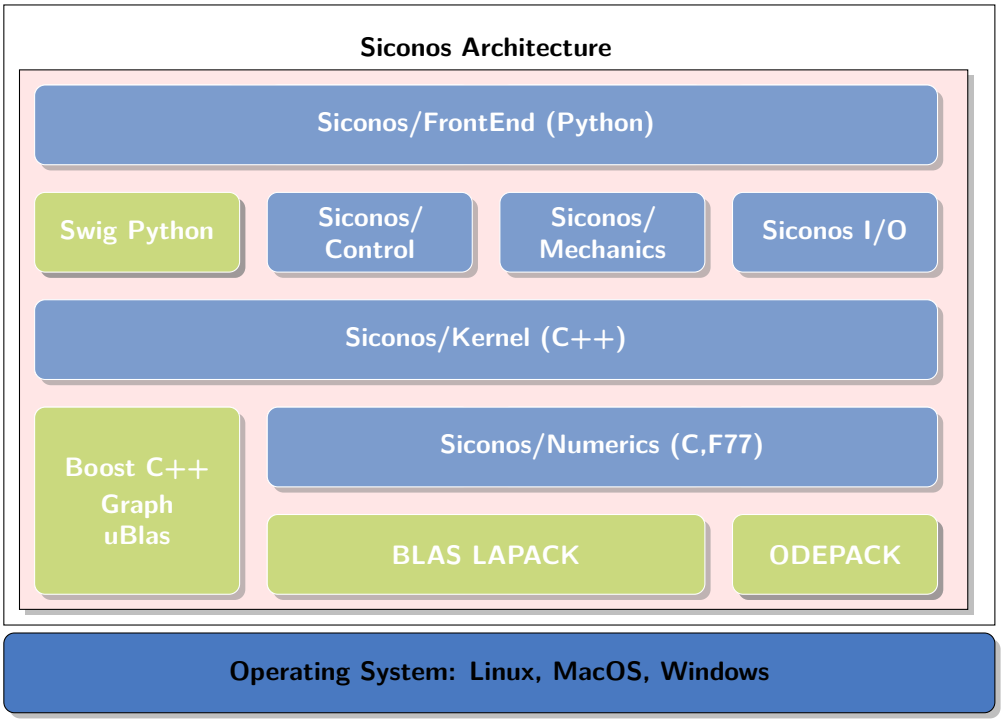


Figure 22: Synopsis of the Siconos libraries

The **Siconos/Kernel** is the core of the software, providing high level description of the studied systems and numerical solving strategies. It is fully written in C++, using extensively the STL and Boost libraries. A complete description of the Kernel is given in Section 16.

The **Siconos/Numerics** part holds all low-level algorithms, to compute basic well-identified problems (ordinary differential equations, LCP, QP ...) and is based on BLAS/LAPACK linear algebra routines and ODEPACK. It is written in C and FORTRAN.

The last component, **Siconos/Front-End**, provides interfaces with the Python language. This to supply more pleasant and easy-access tools for users, during pre/post–treatment.

Note that while the Kernel cannot work without Numerics, Front-End is only an optional pack.

16 Siconos Kernel Components

The Kernel is the central and main part of the Software. The whole dependencies among Kernel parts are fully depicted on Figure 23 below.

All the Kernel implementation is based on the logic we gave in Section 12. It is mainly composed of two rather distinct parts, modeling and simulation, that hold all the objects used respectively in the NSDS modeling (see Section 12.1) and the Simulation description (see Section 12.2).

The Utils module contains tools, mainly to handle classical objects such as matrices or vectors

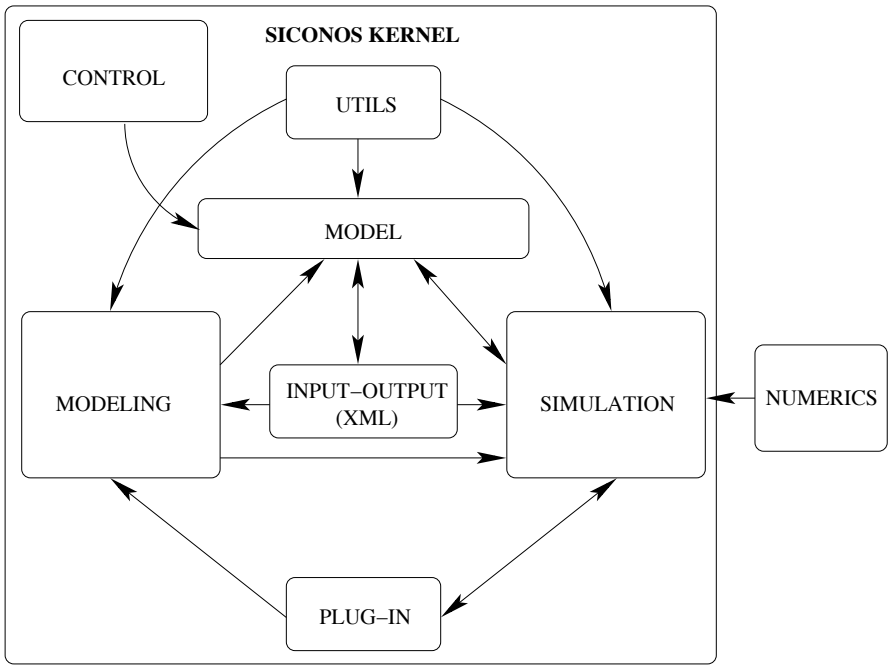


Figure 23: Kernel components dependencies.

and is based on the Boost library[11], especially, uBLAS[12], a C++ library that provides BLAS functionalities for vectors, dense and sparse matrices.

Control package provides objects like `Sensor` and `Actuator`, to add control of the dynamical systems through the `Model` object, as explained in Section 12.3.

A plug-in system is available, mainly to allow the user to provide his own computation methods for some specific functions (vector field of a dynamical system, mass ...), this without having to re-compile the whole platform. Moreover, the platform is designed in a way that allows user to add dedicated modules through object registration and object factories mechanisms (for example to add a specific nonsmooth law, a user-defined sensor ...).

To conclude, class diagrams for modeling and simulation components are given in Figures 24 and 25, which make clearer the various links between all the objects presented before.

Note that the above presentation is only an overall view which is moreover likely to change, so if you are interested in, we encourage you to check on the `http://siconos.gforge.inria.fr/` pages for the last updates.

[11] `http://www.boost.org`
[12] `http://www.boost.org/libs/numeric/ublas/doc/index.htm`

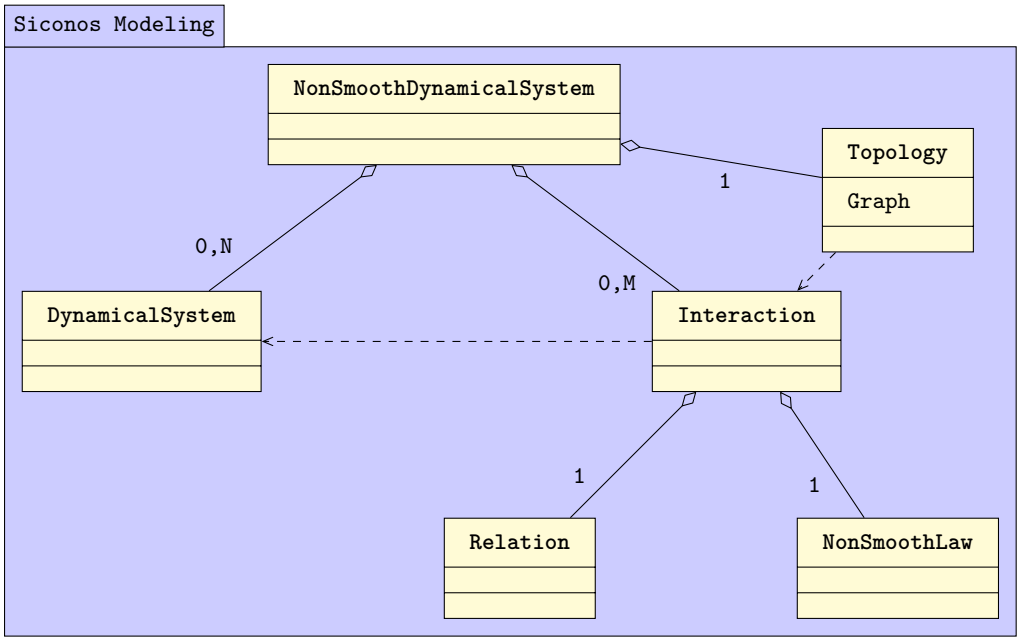


Figure 24: Simplified class diagram for Kernel modeling part.

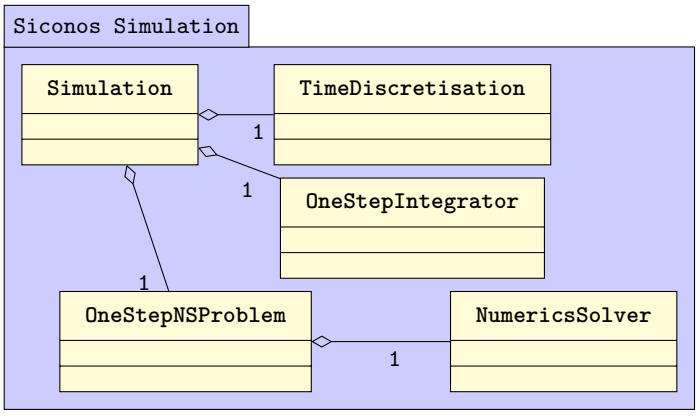


Figure 25: Simplified class diagram for Kernel simulation part.

17 Mechanics engine

In order to formulate general rigid-body mechanical systems, two principle additions are needed: a method to represent joint constraints by means of Relations, and a connection with a collision engine to identify and, since it is impossible to know in advance all possible contact points between a general set of bodies, to manage the lifetimes of temporary 3D friction-contact interactions.

17.1 Joint constraints

The need for joint constraints arises from the present formulation that rigid bodies are represented as Newton-Euler dynamical systems associated with contact geometry. In the sweeping process, constraint violations must be identified and rectified by the numerical solver. Articulations are thus represented by equalities defining manifolds within the Newton-Euler coordinate system in which movement must take place, and the numerical solver's job is to calculate the forces needed to maintain this invariant.

Additionally, a useful mechanics engine will also provide means to measure quantities in the degrees of freedom of the articulations, for example to measure valid movement or to provide feedback such as PID control external to the simulation; this is important in applications involving robotic simulation for example, where the simulation plays the role of a real mechanism that must measure its own displacement and react to its state with respect to its environment.

Therefore, in the joint formulation, a `Relation` class `NewtonEulerJointR` provides a common interface for measuring the position and first derivative of each of its degrees of constraint and degrees of freedom. In our implementation the derivatives are pre-calculated using the symbolic engine `sympy`.

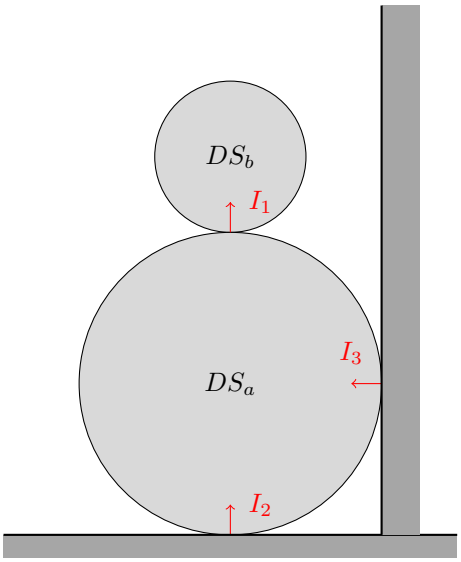
Figure 26: TODO UML diagram for NewtonEulerJointR and family.

17.2 Collision constraints

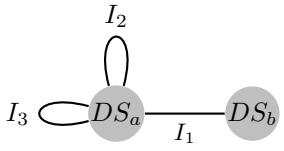
Since the problem of collision detection is a separate problem domain to the simulation issues that Siconos addresses, we delegate collision queries to an external software.

Currently, this is principally done with the collision sub-library of the open source software Bullet Physics, written by Eric Coumans, which is used quite commonly in the video game and simulation industry. However, Siconos implements an implementation-independent interface such that this can in principle be swapped with other collision engines. Currently work is underway for interfacing with the OpenCascade computational geometry engine.

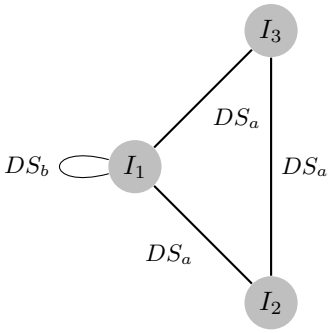
The only requirements of a collision engine to work with Siconos are the ability to identify potential contact points, ideally, within a step or two before contact is actually made, and to be able to stably track these through several timesteps. Bullet manages this by defining a user-defined "margin" around objects which is used as the minimum distance for the GJK broadphase collision query based on the Minkowsky difference of the surrounding convex hulls [34].



(a) a mechanical example



(b) dynamical systems graph



(c) interactions graph

18 Miscellaneous ... find a proper title!

Is it necessary to write something about the following points? To be discussed.

- boost graph
- shared pointers and their friend
- xml python
- boost ublas
- test suite
- autodocumentation

19 Download and installation

The software can be downloaded at `http://siconos.gforge.inria.fr/`, where one can also find an installation guide, a tutorial, the full Doxygen documentation of the code, support, mailing lists and all that sort of utilities.

Part V
Examples

20 Mechanical examples

REMPLACER BRAS ROBOT par exemple de COnstantin Ajouter disques Maurice et Fc3D de Houari

20.1 A column of beads

We consider a column of 1000 spherical beads, in contact or not, falling down to the ground. The modeling is quite the same as for the single ball, but needs the definition of more dynamical systems, one for each bead, and more interactions, one for each potential contact between two beads. The interest of this example lies in the important number of degrees of freedom (i.e.size of vector q) and of relations (size of y and λ) equal to 1000. Figure 27 displays vertical displacements of the 8 lowest beads according to time.

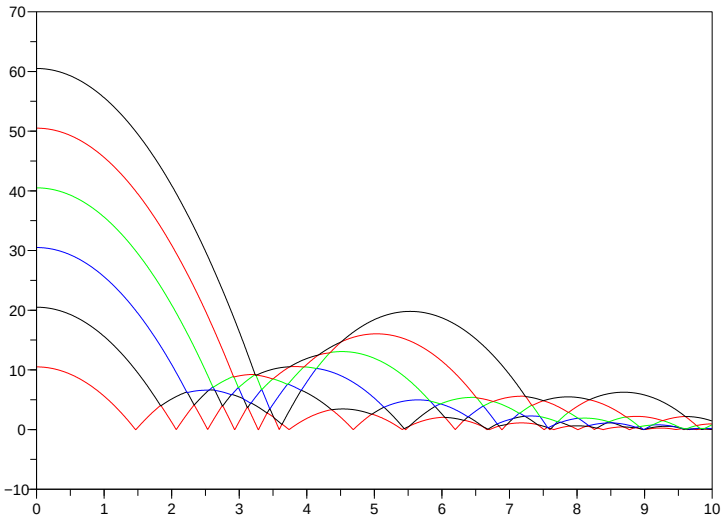


Figure 27: Vertical displacement of the ten lowest beads according to time.

20.2 A Lagrangian Nonlinear System: Simulation of a Robotic Arm

`XXX replace by Rover ESA simulation` As a second example, we consider now the Mitsubishi PA-10 Robot, a seven degree–of–freedom anthropomorphic robot, presented on Figure 28.

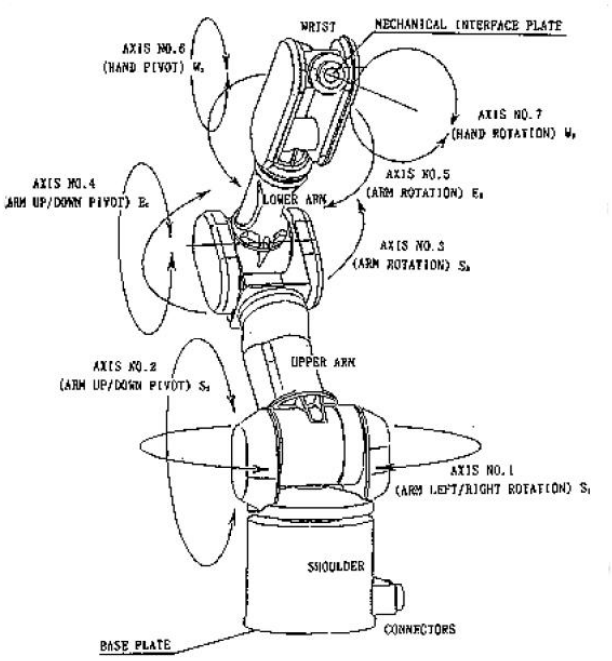


Figure 28: Mitsubishi PA-10 robotic arm

In Siconos, this robotic arm is modeled using a Lagrangian (nonlinear) dynamical system, that interacts with the ground. The arm falls down due to its own weight and bounces on the ground. Moreover, articular stops have been included to limit angular rotations. The degrees of freedom are the rotation angles, represented by the vector q. Neither external forces nor control terms are present. Finally, the robot dynamics is cast into the framework of Lagrangian dynamical systems as in (7). As a relation, we set that the distance between the arm and the ground must remain positive, which means that contacts can occur but without penetration. The contact is supposed to be frictionless with a restitution coefficient denoted as e. This leads to nonlinear links between the angular position (i.e. components of q), that can be written as:

$$\dot{y} = \nabla_q h(q)\dot{q} \quad \text{and} \quad r = g(q)\lambda = \nabla_q^t h(q)\lambda$$

A Newton-Impact non smooth law complete this set of equations.

$$\dot{y}(t^+) = -e\dot{y}(t^-)$$

with a restitution coefficient $e = 0.9$ for contact with the ground and $e = 0$ for angular stops.

The results of Siconos simulation, using Moreau time-stepping, Newton algorithm and a non smooth quadratic programming solver, are presented on Figures 29 and 30. We denote A and B

Figure 29: robotic arm fall-down (a) initial position - (b) before contact with ground - (c) contact - (d) post-contact

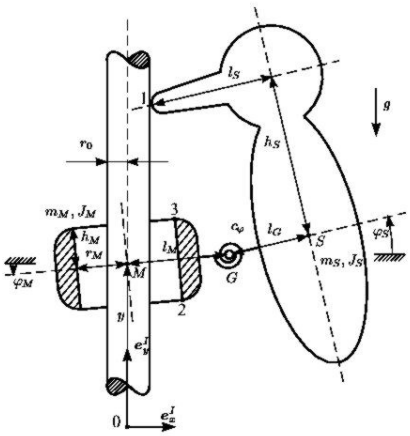
respectively the first (the one clamped on the ground) and second parts of the arm; θ_1 is the angle between A and the vertical position and θ_2 the angle between A and B. First, θ_1 and θ_2 increase, until the last one reaches its maximum angular position. Then θ_1, goes on increasing and θ_2 remains constant until the extremity of the arm touches the ground and bounces, here with a restitution coefficient e equal to 0.9. Finally A touches the ground and bounces also, together with B, until the complete arm lies on the ground. Note on 30.c and d that the angular velocity cancels when the angular stops are reached, and change their signs when contact is established.

Figure 30: Siconos simulation of the robotic arm

20.3 The Woodpecker Toy

This system has been implemented in Siconos by M. Moeller from ETH Zurich, following the examples proposed in [47]. The Woodpecker toy is presented on Figure 31.a and b and consists in a sleeve, a spring and the woodpecker. The hole in the sleeve is slightly larger than the diameter of the pole, thus allowing a kind of pitching motion interrupted by impacts with friction. Its dynamical behavior shows both impact and friction phenomena.

(a)



(b)

Figure 31: The woodpecker toy.

The Woodpecker Toy is a system which can only operate in the presence of friction as it relies on combined impacts and jamming. Among other things, an animation of the toy can be found there: `http://www.zfm.ethz.ch/ leine/toys.htm`.

Some results obtained with Siconos are presented on Figure 32.

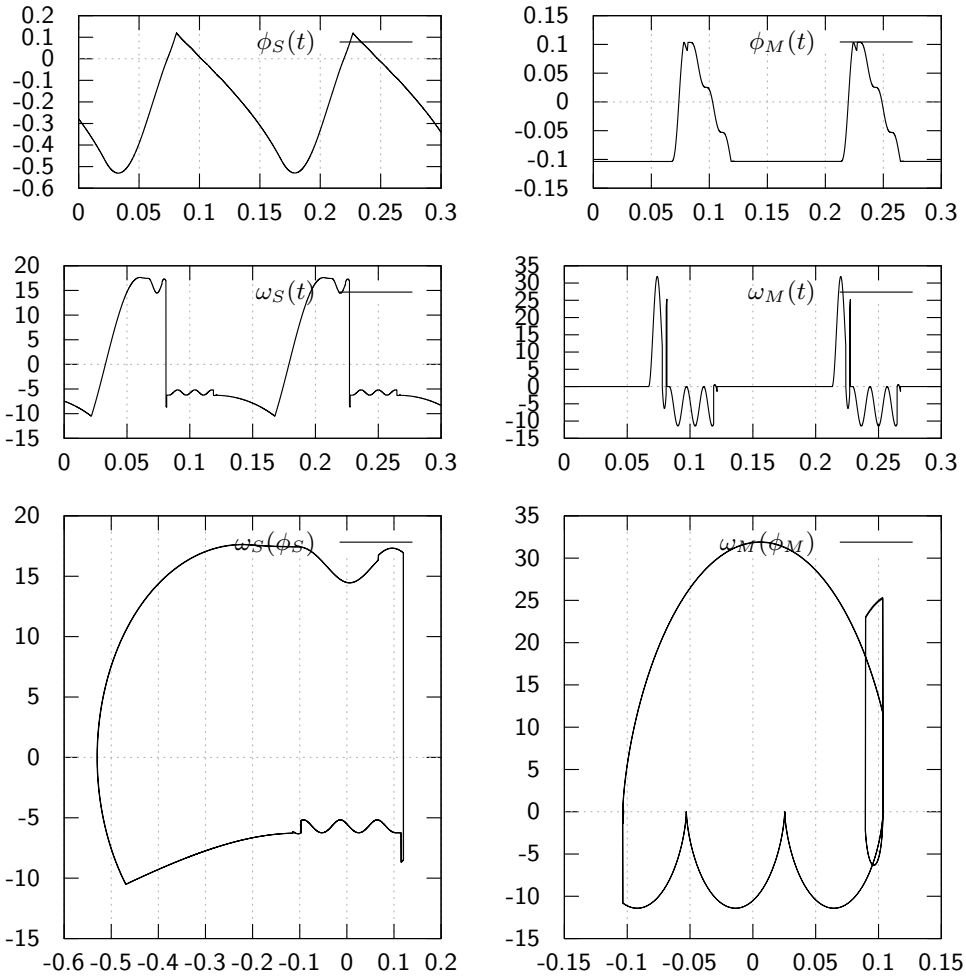


Figure 32: Simulation results for the woodpecker toy using Siconos platform.

20.4 The Cam Follower System

The cam-follower system has been proposed and implemented by G. Osorio, Mario di Bernardo and Stefania Santini from University of Naples Federico II, Italy.

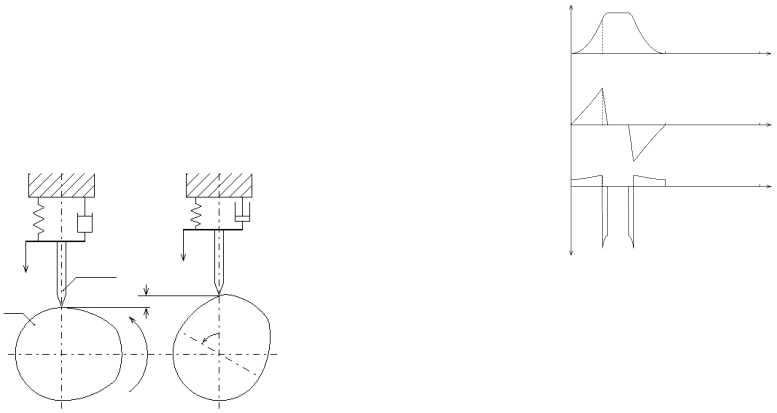
The cam-follower system is represented on Figure 33. The free body dynamics can be described by a linear second order system. An external input is considered acting directly on the follower. This input is a nonlinear forcing component coming from the valve. The follower motion is constrained to a phase space region bounded by the cam position. The non conservative Newton restitution law is used for the computation of the post impact velocity. The cam is assumed to be massive, therefore only rotational displacement is allowed. Under these assumptions, the free body dynamics of the follower can be described by:

$$\mu\frac{d^2u(t)}{dt^2} + \zeta\frac{du(t)}{dt} + \kappa u(t) = f_v(t),\ if\ u(t) > c(t).$$

where μ, ζ and κ are constant parameters for the follower mass, friction viscous damping and spring stiffness respectively. The state of the follower is given by the position $u(t)$ and velocity $v(t) = \frac{du}{dt}$. The external forcing is given by $f_v(t)$. The cam angular position determines $c(t)$ that defines the holonomic (i.e. constraint only on the position) rheonomic (i.e. time varying) constraint. The dynamic behavior when impacts occurs (i.e. $u(t) = c(t)$) is modelled via Newton's impact law that in this case is given by this case is given by:

$$v(t^+) = \frac{dc}{dt} - r\left(v(t^-) - \frac{dc}{dt}\right) = (1+r)\frac{dc}{dt} - rv(t^-),\ if\ u(t) = c(t).$$

where $v(t^+)$ and $v(t^-)$ are the post and pre impact velocities respectively, $\frac{dc}{dt}$ is the velocity vector of the cam at the contact point with the follower, and $r \in [0, 1]$ is the restitution coefficient to model from plastic to elastic impacts. In Figure 33 is presented the schematic diagram of the physical cam-follower system. In Figure 33.a for $t = 0$, 1.b for $t = \beta$, and 33.c the profile of the constraint position $\delta c(t)$, velocity $\frac{dc}{dt}(t)$ and acceleration $\frac{d^2c}{dt^2}(t)$. It is possible to visualize the follower displacement as a function of the cam position. It is also important to notice that different types of cams and followers profiles are used in practical applications.



(a) (b)

Figure 33: (a) Schematic diagram of the cam-follower system - (b) profiles of the constraint position.

It is possible to completely describe the cam-follower system as a driven impact oscillator into the framework of Lagrangian NSDS using a translation in space. Setting $\hat{u}(t) = u(t) - c(t)$ and $\hat{v}(t) = v(t) - dc/dt$, then equations (**??**) and (**??**) can be expressed as (the argument t will not be explicitly written)

$$\mu\frac{d^2\hat{u}}{dt^2}+\zeta\frac{d\hat{u}}{dt}+\kappa\hat{u}=f_v-\left(\mu\frac{d^2c}{dt^2}+\zeta\frac{dc}{dt}+\kappa c\right)\equiv \quad \hat{f} \quad \text{if } \hat{u}>0$$
$$\hat{v}^+ = -r\hat{v}^- \quad \text{if } \hat{u}=0.$$

Using the framework presented in (7) we can derive all of the terms which define a Lagrangian NSDS. In our case the model is completely linear:

$$q=\left[\begin{array}{c}\hat{u}\end{array}\right], M(q)=\left[\begin{array}{c}\mu\end{array}\right], \qquad NNL(q,\dot{q})=\left[\begin{array}{c}0\end{array}\right]$$
$$F_{int}(t,q,\dot{q})=\left[\begin{array}{c}\zeta\end{array}\right]\dot{q}+\left[\begin{array}{c}\kappa\end{array}\right]q, F_{ext}=\left[\begin{array}{c}\hat{f}\end{array}\right]$$

The unilateral constraint requires that $\hat{u}\geqslant 0$ so we can obtain

$$y=H^Tq+b, H^T=\left[\begin{array}{c}1\end{array}\right], \qquad b=0$$

In the same way, the reaction force due to the constraint is written as follows:

$$R=H\lambda$$

The unilateral contact law may be formulated as in (**??**) with a complementarity condition and a Newton's impact law.

Simulation The simulation of the cam follower system has been performed for different values of the cam rotational speed with the SICONOS software package using a time-stepping numerical scheme with step size ($h = 1.10^{-4}$) and an event-driven scheme with minimum step size ($h_{min} = 1.10^{-12}$). Figure 34 and 35 show the time simulations for different values of the cam rotational speed and Figure 36 shows the chaotic attractor at $rpm = 660$ for impact and stroboscopic Poincaré sections.

21 Electrical circuits examples

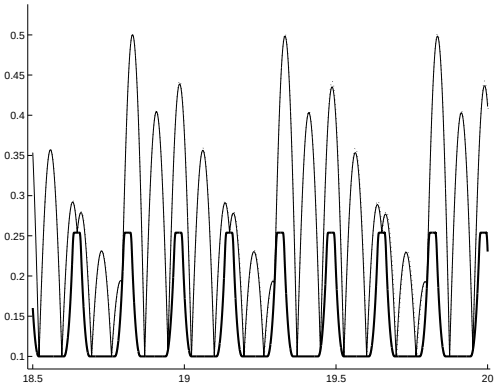
21.1 MOS Transistors and Inverters

Most of this work has been done by P. Denoyelle. More details can be found in [26]

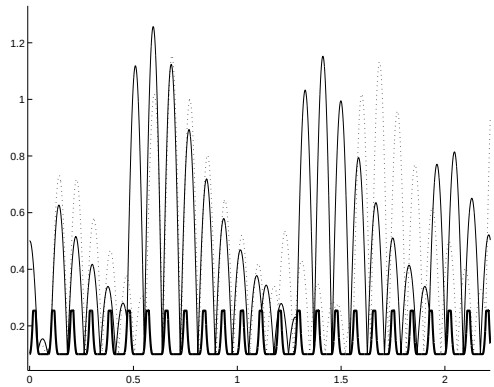
21.1.1 Piecewise Linear Model of a MOS Transistor

People could benefit from a simplification of devices models (e.g MOS models) in the form of a piecewise linear representation instead of the complicated formula implemented in SPICE simulators. For instance, in [46], the authors considered the Sah model of the NMOS static characteristic:

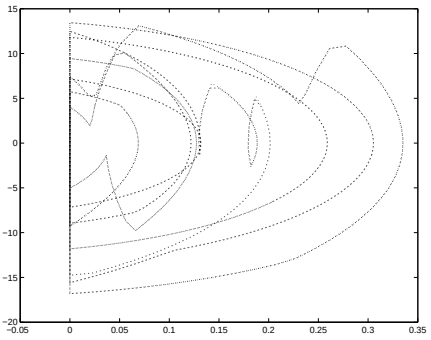
$$I_{DS}=\frac{K}{2}\cdot\left(f(V_G-V_S-V_T)-f(V_G-V_D-V_T)\right)$$


0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
18.5
19
19.5
20

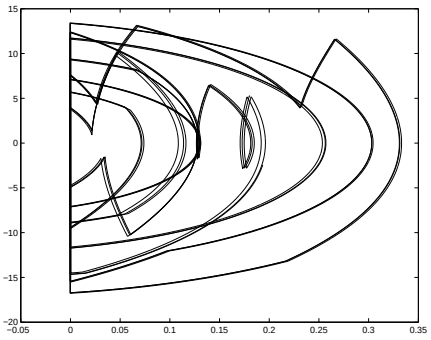
(a)


1.2
1
0.8
0.6
0.4
0.2
0
0.5
1
1.5
2

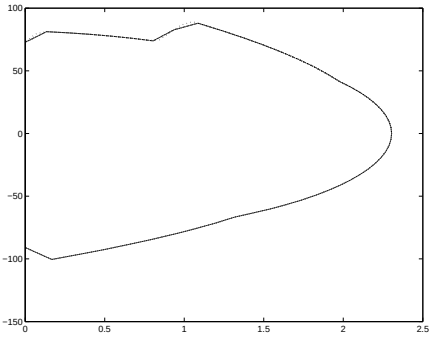
(b)



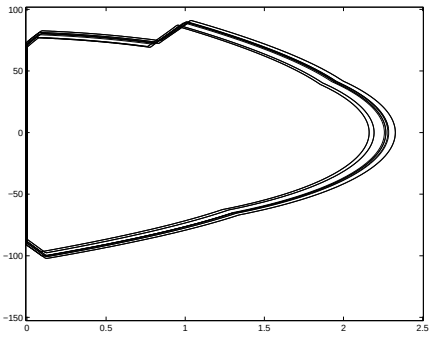
(a)



(b)

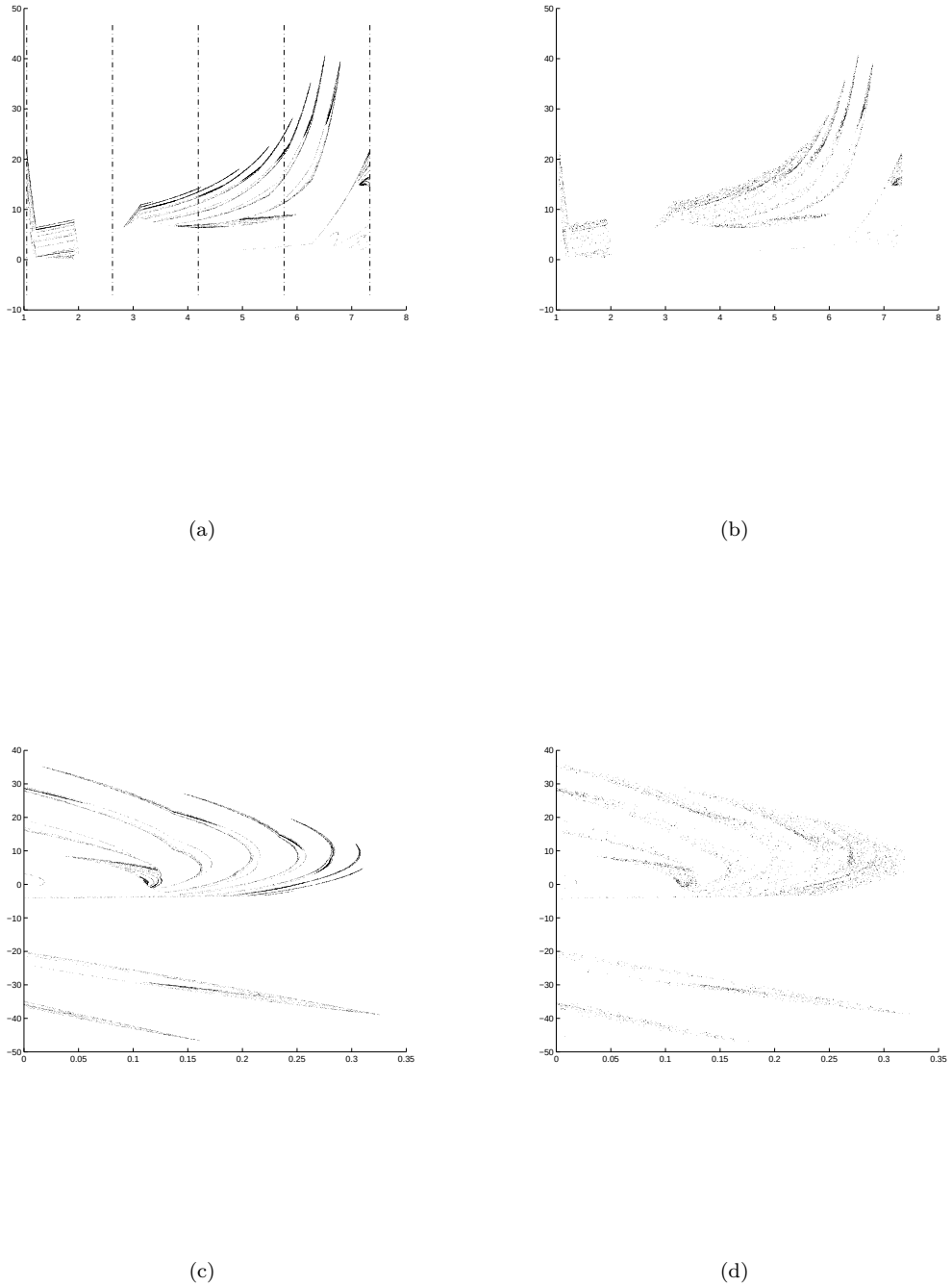


(c)



(d)

Figure 35: The Cam Follower System. State space comparison using SICONOS platform. (a) rpm=358. Event Driven (b) rpm=358. Time Stepping (h=1e-4) (c) rpm=700. Event Driven (d) rpm=700. Time Stepping (h=1e-4).



(a) (b)

(c) (d)

Figure 36: The Cam Follower System. Attractors comparison using SICONOS platform at rpm=660. (a) Impact map. (Event Driven) (b) Impact Map. Time Stepping (h=1e-4)(a) Stroboscopic map. (Event Driven) (b) Stroboscopic Map. Time Stepping (h=1e-4).

with:

$K = \frac{\mu C_{OX} W}{L}$

μ mobility of majority carriers

(sample values of 750 $cm^2.V^{-1}.s^{-1}$ for a NMOS, 250 $cm^2.V^{-1}.s^{-1}$ for a PMOS)

$C_{OX} = \frac{\epsilon_{SiO_2}}{t_{OX}}$

$\epsilon_{SiO_2} = \epsilon_{r\ SiO_2} \cdot \epsilon_0$ ($\epsilon_{r\ SiO_2} \approx 3.9$)

t_{OX} oxide thickness $\approx 4nm$ in a recent $180nm$ technology

W channel width

L channel length $\approx 130nm$ in a recent $180nm$ technology

V_T threshold voltage depending on technology, V_{BS} , temperature ≈ 0.25 to 1 V

The function $f : \mathbb{R} \longrightarrow \mathbb{R}$ is defined as:

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x^2 & \text{if } x \geqslant 0 \end{cases}$$

The piecewise and quadratic nature of this function was approximated by the following 6 segments piecewise linear function by the authors of [46] (see Figure 37):

$$f_{PWL}(x) = \begin{cases} 0 & \text{if } x < 0 \\ 0.09 \cdot x & \text{if } 0 \leqslant x < 0.1 \\ 0.314055 \cdot x - 0.0224055 & \text{if } 0.1 \leqslant x < 0.2487 \\ 0.780422 \cdot x - 0.138391 & \text{if } 0.2487 \leqslant x < 0.6185 \\ 1.94107 \cdot x - 0.856254 & \text{if } 0.6185 \leqslant x < 1.5383 \\ 4.82766 \cdot x - 5.29668 & \text{if } 1.5383 \leqslant x \end{cases}$$

The relative error between f and f_{PWL} is kept below 0.1 for $0.1 \leqslant x < 3.82$. The absolute error is less than $2 \cdot 10^{-3}$ for $0 \leqslant x < 0.1$ and 0 for negative x. In practice, the values of V_G, V_S, V_D, V_T in logic integrated circuits allow a good approximation of f by f_{PWL}.

The figure 38 displays the static characteristic $I_{DS}(V_{GS}, V_{DS})$ of an NMOS obtained with the SPICE level 1 model and the piecewise linear approximation of the Sah model. The following parameter values were used:

$$\epsilon_{r\ SiO_2} = 3.9$$
$$t_{OX} = 20\ nm$$
$$\mu = 750\ cm^2.V^{-1}.s^{-1}$$
$$W = 1\ \mu m$$
$$L = 1\ \mu m$$
$$V_T = 1\ V$$

Bottom figures include both models results with two different viewpoints to display the regions where differences appear.

21.1.2 Inverter Chain

This simple model of a NMOS transistor was adapted to the PMOS transistor and both models were used to simulate an inverter chain (see Figure 39). The output of each inverter is loaded by the intrinsic capacitances of transistors (with values of a few fF) and a load capacitor of 50 fF representing the wiring between successive inverters.

In these early simulations, the dynamical behavior of the MOS transistor was simplified by keeping the intrinsic capacitances C_{GS} and C_{GD} independent from voltages. Of course, this differs from the Meyer nonlinear capacitances implemented in the SPICE level 1 model. Figure 40 shows the comparison between simulation results with SPICE (dotted lines) and SICONOS for a selection of inverters output voltage and MOS currents.

21.2 Power Converters

The functioning of switch mode power supplies is based on flyback diodes and switched transistors with a regulation on output current or voltage. The article [15] presents some simulation results of a Cuk converter [13] which have been reproduced in Siconos.

This example has been proposed and developed by I. Merillas, E. Fossas and Carles Battle from Universitat Politècnica de Catalunya (UPC), Barcelona [50]. The system consists in a Parallel Resonant Converter (PRC) which is basically a dc-dc power converter. The schematic of the PRC is shown in Figure 41. The system is composed of four parts: an inverter block, a resonant tank in series, a rectifier block and an output filter. In our case, the inverter block is a full-bridge inverter. It is called parallel resonant converter because the load is in parallel with the resonant capacitor.

To represent the system in Siconos, we use a `FirstOrderLinearDS`, a `FirstOrderLinearTIR` and a `ComplementarityConditionNSL`, as given in (6), (8) and (**??**) respectively.

[13] This work was supported by the European project SICONOS IST-2001-37172.

An thus we obtain a LCS (Linear Complementarity System) of the form:

$$\begin{aligned} \dot{x}(t) &= Ax(t) + b(t) + r, \\ y(t) &= Cx(t) + D\lambda(t), \\ 0 &\leqslant y \perp \lambda \geqslant 0 \end{aligned}$$

with the state $x = [i_r, v_r, i_L, v_0]$, and the complementarity variables $y = [v_{D1}, v_{D3}, i_{D2}, i_{D4}]$ and $\lambda = [i_{D1}, i_{D3}, v_{D2}, v_{D4}]$. And:

$$A = \begin{pmatrix} 0 & -\frac{1}{L_r} & 0 & 0 \\ \frac{1}{C_r} & 0 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{L_f} \\ 0 & 0 & \frac{1}{C_f} & -\frac{1}{R_L C_f} \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 & 0 & 0 \\ -\frac{1}{nC_r} & \frac{1}{nC_r} & 0 & 0 \\ 0 & 0 & \frac{1}{L_f} & \frac{1}{L_f} \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

$$C = \begin{pmatrix} 0 & -\frac{1}{n} & 0 & 0 \\ 0 & \frac{1}{n} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, D = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix},$$

And for the external source:

$$b(t) = E\,\mathrm{Sign}(sin(wt)), E = \begin{pmatrix} \frac{1}{L_r} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

The numerical simulations are performed with the following parameter values: $L_r = 150\mu H$, $L_f = 0.4mH$, $C_r = 68nF$, $C_f = 2.2\mu F$, $R_L = 33\Omega$ and the frequency of the input voltage $F_0 = 55000Hz$. The results are given on Figure 42.

22 Gene regulatory networks.

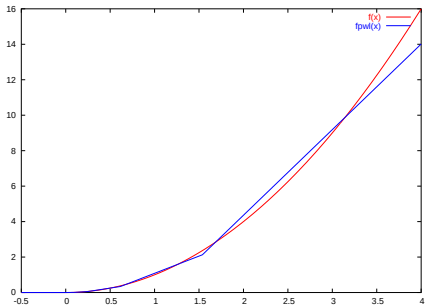


Figure 37: Piecewise linear approximation of f

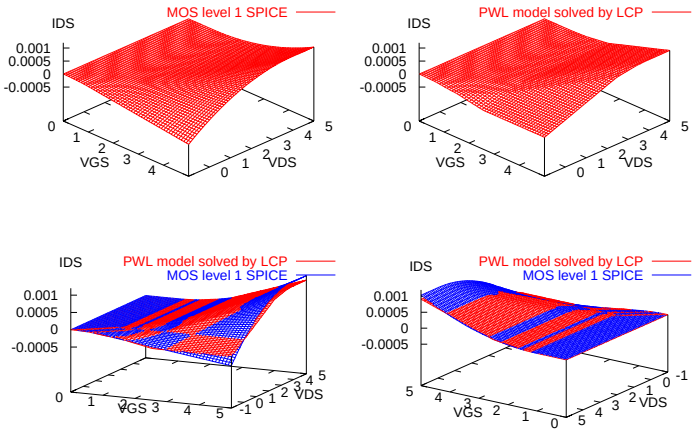


Figure 38: Static characteristic of an NMOS transistor with a simple PWL model and SPICE level 1 model

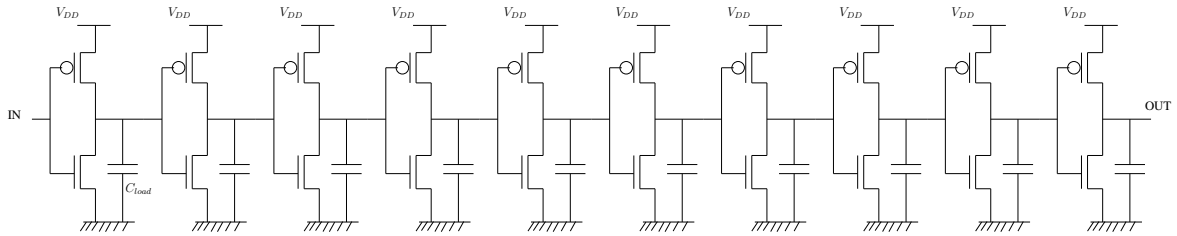
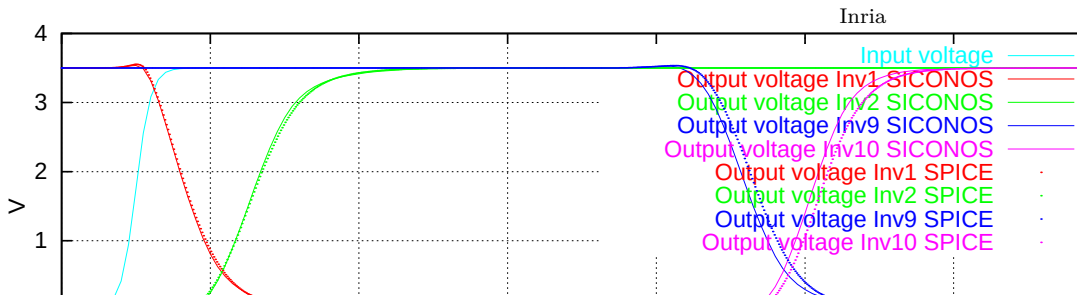


Figure 39: Inverter chain in CMOS


Inria
4
3
2
1
V
Input voltage
Output voltage Inv1 SICONOS
Output voltage Inv2 SICONOS
Output voltage Inv9 SICONOS
Output voltage Inv10 SICONOS
Output voltage Inv1 SPICE
Output voltage Inv2 SPICE
Output voltage Inv9 SPICE
Output voltage Inv10 SPICE

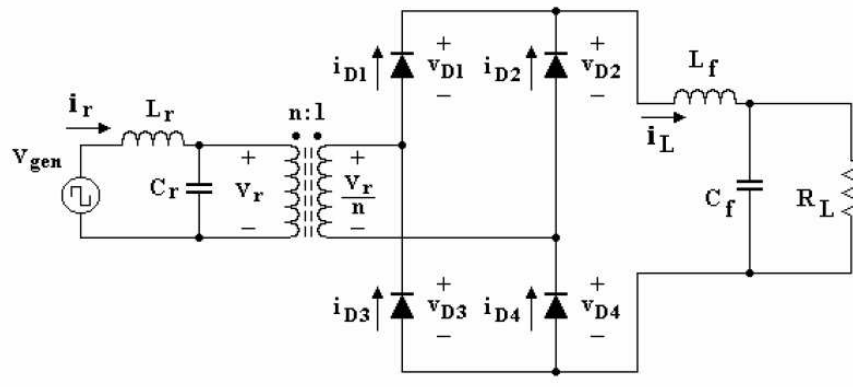
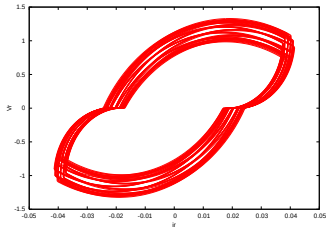
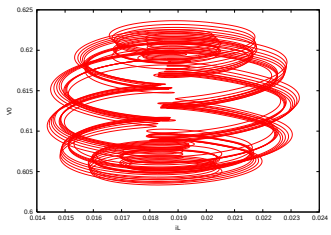


Figure 41: A Parallel Resonant Converter diagram



(a) V_r according to i_R



(b) V_0 according to i_L

Figure 42: Siconos simulation of the parallel resonant converter, with $R_L = 33\Omega$ and $F0 = 55KHz$.

Acknowledgments

This work was supported by the European Community through the Information Society Technologies thematic program under the project SICONOS (IST-2001-37172). Originally, SICONOS was an European Project, starting in September 1, 2002, ending in December 31, 2006, gathering scientists from various communities like Mechanics, Applied Mathematics, Systems and Control, and Numerical Analysis. More details are available at `http://siconos.inrialpes.fr`.

The project has resulted in the Siconos Platform, a scientific computing software dedicated to the modeling, simulation, control and analysis of NonSmooth Dynamical Systems (NSDS), mainly developed in the BipOp team (`http://bipop.inrialpes.fr`) at INRIA Rhônes-Alpes in Grenoble and distributed under GPL GNU license.

This work has also been supported by the French National Research Agency (ANR) through COSINUS program (project SALADYN ANR-08-COSI-014)`http://saladyn.inria.gforge.fr/`.

The authors want to thank F. Dubois and R. Pissard-Gibollet for their great help in the design and the implementation of the Siconos platform and for their constant support. M. Jean has also to be acknowledged as a pioneer in simulation of nonsmooth dynamical systems with open–source software LMGC and LMGC90 [27].

The authors acknowledge the Siconos partners who have contributed the examples section of this document, especially, Mathias Moeller for his contribution to the woodpecker toy, Mario di Bernardo, Gustavo Osorio and Stefania Santini for their contribution to the Cam-follower system, P. Denoyelle for the Mos transistor modelling and I. Merillas for the power converter simulation.

Finally, the discussion and the feedback of many colleagues have to be acknowledged : M.Jean, M. Renouf, R. Mozul, S. Mazet, A. Tanwani, S. Nguyen, J. Michalczyk, H.H. Do Nguyen, R. Kikuuwe, ...

Infos sur le projet europeen ? Remerciements ? Liens ou autres infos pratiques pour la plate-forme

References

[1] V. Acary. *Dynamics and Control of Switched Electronic Systems Vasca, Francesco; Iannelli, Luigi (Eds.)*, chapter 14. Time-Stepping via Complementarity, pages 417–450. Advances in Industrial Control. Springer Verlag, 2012. XIV, 492 p. 240 illus., 61 in color.

[2] V. Acary, O. Bonnefon, and B. Brogliato. Time-stepping numerical simulation of switched circuits with the nonsmooth dynamical systems approach. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 29(7):1042 – 1055, 2010.

[3] V. Acary, O. Bonnefon, and B. Brogliato. *Nonsmooth Modeling and Simulation for Switched Circuits*, volume 69 of *Lecture Notes in Electrical Engineering*. Springer Verlag, 2011.

[4] V. Acary and B. Brogliato. Concurrent multiple impacts modelling - case-study of a 3-ball chain. In K. J. Bathe, editor, *Second MIT conference on computational Fluid and Solid Mechanics*, pages 1842–1847. Elsevier, jun 2003.

[5] V. Acary and B. Brogliato. *Numerical methods for nonsmooth dynamical systems. Applications in mechanics and electronics.* Lecture Notes in Applied and Computational Mechanics 35. Berlin: Springer. xxi, 525 p. , 2008.

[6] V. Acary and B. Brogliato. Implicit Euler numerical scheme and chattering-free implementation of sliding mode systems. *Systems & Control Letters / Systems and Control Letters*, Provisionally accepted. Available as technical report at http://hal.inria.fr/inria-00374840:0–18, 2009.

[7] V. Acary and B. Brogliato. Implicit euler numerical scheme and chattering-free implementation of sliding mode systems. *Systems and Control Letters.*, 2010. doi:10.1016/j.sysconle.2010.03.002.

[8] V. Acary, B. Brogliato, and D. Goeleven. Higher order Moreau's sweeping process: mathematical formulation and numerical simulation. *Mathematical Programming Ser. A*, 113(1):133–217, 2008.

[9] V. Acary, B. Brogliato, and Y.V. Orlov. Chattering-free digital sliding-mode control with state observer and disturbance rejection. *Automatic Control, IEEE Transactions on*, 57(5):1087 –1101, may 2012.

[10] V. Acary, F. Cadoux, C. Lemaréchal, and J. Malick. A formulation of the linear discrete coulomb friction problem via convex optimization. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 91(2):155–175, 2011.

[11] V. Acary, H. de Jong, and B. Brogliato. Numerical simulation of piecewise-linear models of gene regulatory networks using complementarity systems theory. *Physica D*, 269:103–199, January 2013.

[12] U.M. Ascher and L.R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential–Algebraic Equations*. SIAM, 1998.

[13] P. Ballard. The dynamics of discrete mechanical systems with perfect unilateral constraints. *arma*, 154:199–274, 2000.

[14] P. Ballard. Bounded variations and measure theory on the line. Lectures notes of the second summer on Nonsmooth Dynamics held in Autrans (France), 2003.

[15] C. Batlle, E. Fossas, I. Merillas, and A. Miralles. Generalized discontinuous conduction modes in the complementarity formalism. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 52, no.8:447–451, Aug. 2005.

[16] O.A. Bauchau and A. Laulusa. Review of contemporary approaches for constraint enforcement in multibody systems. *Journal of Computational and Nonlinear Dynamics*, 3(1):011005, 2008.

[17] K.E. Brenan, S. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. North-holland, 1989.

[18] B. Brogliato. *Nonsmooth Mechanics: Models, Dynamics and Control*. Communications and Control Engineering. Springer-Verlag, London, 2nd edition, 1999.

[19] B. Brogliato. Absolute stability and the Lagrange-Dirichlet theorem with monotone multivalued mappings. *Systems and Control Letters*, 51:343–353, 2004.

[20] B. Brogliato and D. Goeleven. Well-posedness, stability and invariance results for a class of multivalued lur'e dynamical systems. *Nonlinear Analysis: Theory, Methods and Applications*, 74:195–212, 2011.

[21] B. Brogliato and L. Thibault. Well-posedness results for non-autonomous complementairy systems. *Journal of Convex Analysis*, 17(3-4):961–990, 2010.

[22] M.K. Camlibel, W.P.M.H. Heemels, and J.M. Schumacher. On linear passive complementarity systems. *European Journal of Control*, 8:220–237, 2002.

[23] F.H. Clarke. *Optimization and Nonsmooth analysis*. Wiley, New York, 1983.

[24] R. W. Cottle, J. Pang, and R. E. Stone. *The linear complementarity problem*. Academic Press, Inc., Boston, MA, 1992.

[25] G. De Saxcé and Z.-Q. Feng. New inequality and functional for contact with friction : The implicit standard material approach. *msm*, 19(3):301–305, 1990.

[26] P. Denoyelle and V. Acary. The non-smooth approach applied to simulating integrated circuits and power electronics. Evolution of electronic circuit simulators towards fast-SPICE performance. *INRIA Research Report 0321, http://hal.inria.fr/docs/00/08/09/20/PDF/RT-0321.pdf*, 2006.

[27] F. Dubois and M. Jean. LMGC90 une plateforme de développement dédiée à la modélisation des problèmes d'interaction. In *Actes du sixième colloque national en calcul des structures*, volume 1, pages 111–118. CSMA-AFM-LMS, 2003.

[28] H. Elmqvist, S.E. Mattsson, and M. and Otter. Object-oriented and hybrid modeling in modelica. *Journal Européen des systèmes automatisés*, 35(4):395 – 404, 2001.

[29] Francisco Facchinei and Jong-Shi Pang. *Finite-dimensional variational inequalities and complementarity problems*, volume I & II of *Springer Series in Operations Research*. Springer Verlag NY. Inc., 2003.

[30] A. F. Filippov. *Differential equations with discontinuous right hand sides*. Kluwer, Dordrecht, the Netherlands, 1988.

[31] R. Fletcher. *Practical Methods of Optimization*. Chichester: John Wiley & Sons, Inc., 1987.

[32] C. Georgescu, B. Brogliato, and V. Acary. Switching, relay and complementarity systems: A tutorial on their well-posedness and relationships. *Physica D: Nonlinear Phenomena*, 241(22):1985 – 2002, 2012. Dynamics and Bifurcations of Nonsmooth Systems.

[33] M. Géradin and A. Cardona. *Flexible Multibody Dynamics: A finite element Approach*. J. Wiley & Sons, New York, 2001. 340 p.

[34] Elmer G Gilbert, Daniel W Johnson, and S Sathiya Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.

[35] C. Glocker. *Set-Valued Force Laws: Dynamics of Non-Smooth systems*, volume 1 of *Lecture Notes in Applied Mechanics*. Springer Verlag, 2001.

[36] C. Glocker. Concepts for modeling impacts without friction. *Acta Mechanica*, 168(1–2):1–19, 2004.

[37] H. Goldstein. *Classical Mechanics*. Addison-Wesley Pub. Co., 2nd edition, 1980.

[38] E. Hairer and G. Wanner. *Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems.*, volume 14 of *Series in Comput. Mathematics*. Springer, second revised edition, 1996.

[39] W.P.M.H. Heemels, J.M. Schumacher, and S. Weiland. Linear complementarity systems. *SIAM J. Appl. Math*, 60:1234–1269, 2000.

[40] Olivier Huber, Vincent Acary, and Bernard Brogliato. Comparison between explicit and implicit discrete-time implementations of sliding-mode controllers. In *CDC 2013 - 52nd IEEE Conference on Decision and Control*, Florence, Italy, October 2013.

[41] Olivier Huber, Vincent Acary, Bernard Brogliato, and Franck Plestan. Discrete-time twisting controller without numerical chattering: analysis and experimental results with an implicit method. In *CDC 2014 - IEEE 53rd Annual Conference on Decision and Control*, Los Angeles, United States, September 2014. IEEE.

[42] O. Janin and C.H. Lamarque. Comparison of several numerical methods for mechanical systems with impacts. *Int. J. Numer. Methods Eng.*, 51(9):1101–1132, 2001.

[43] M. Kunze and M.D.P. Monteiro Marquès. An introduction to Moreau's sweeping process. In B. Brogliato, editor, *Impact in Mechanical systems: Analysis and Modelling*, volume 551 of *Lecture Notes in Physics*, pages 1–60. Springer, 2000.

[44] D.M.W. Leenaerts. On linear dynamic complementarity systems. *IEEE Transactions on Circuits and Systems*, 46(8):1022–1026, 1999.

[45] D.M.W. Leenaerts and W.M. Van Bokhoven. *Piecewise Linear Modeling and Analysis*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[46] Domine M. W. Leenaerts and Wim M. Van Bokhoven. *Piecewise Linear Modeling and Analysis*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[47] Ch. Leine, R. Glocker and D.H. Van Campen. Nonlinear dynamics and modelling of various wooden toys with impact and friction. *Journal of Vibration and Control,*, 9:25–78, 2001.

[48] R. Leine and H. Nijmeijer. *Dynamics and Bifurcations of Non-Smooth Mechanical Systems*. Springer Verlag, Lecture Notes in Applied and Computational Mechanics 18, 2004.

[49] M. Mabrouk. A unified variational model for the dynamics of perfect unilateral constraints. *Eur. J. Mecha. A Solids*, 17(5):819–842, 1998.

[50] I. Merillas Santos. *Modeling and Numerical Study of Nonsmooth Dynamical systems*. PhD thesis, Universitat Politècnica de Catalunya, December 2006.

[51] M.D.P. Monteiro Marques. *Differential Inclusions in Nonsmooth Mechanical Problems. Shocks and Dry Friction*. Progress in Nonlinear Differential Equations and their Applications, vol.9. Birkhauser, Basel, 1993.

[52] J.J. Moreau. Liaisons unilatérales sans frottement et chocs inélastiques. *cras*, 296 série II:1473–1476, 1983.

[53] J.J. Moreau. Standard inelastic shocks and the dynamics of unilateral constraints. In G. Del Piero and F. Maceri, editors, *Unilateral Problems in Structural Analysis*, number 288 in CISM Course and Lectures, pages 173–221. Springer Verlag, 1985.

[54] J.J. Moreau. Unilateral contact and dry friction in finite freedom dynamics. In J.J. Moreau and P.D. Panagiotopoulos, editors, *Nonsmooth mechanics and applications*, number 302 in CISM, Courses and lectures, pages 1–82. CISM 302, Spinger Verlag, Wien- New York, 1988. Formulation mathematiques tire du livre Contacts mechanics.

[55] J.J. Moreau. Numerical aspects of the sweeping process. *cmame*, 177:329–349, 1999. Special issue on computational modeling of contact and friction, J.A.C. Martins and A. Klarbring, editors.

[56] Ngoc Son Nguyen and Bernard Brogliato. *Multiple Impacts in Dissipative Granular Chains*, volume 72 of *Lecture Notes in Applied and Computational Mechanics*. Springer Verlag, 2014. XXII, 234 p. 109 illus.

[57] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Verlag, 1999.

[58] J.-S. Pang and D. Stewart. Differential variational inequalities. *Mathematical Programming A*, 113(2), 2008.

[59] J.S. Pang and R. Chandrasekaran. Linear complementarity problems solvable by a polynomially bounded pivoting algorithms. *Math. Prog. Study*, 25:13–27, 1985.

[60] F. Pfeiffer and C. Glocker. *Multibody Dynamics with Unilateral Contacts*. Non-linear Dynamics. John Wiley & Sons, 1996.

[61] A. Y. Pogromski, W.P.M.H. Heemels, and H. Nijmeijer. On solution concepts and well-posedness of linear relay systems. *Automatica*, 39:2139–2147, 2003.

[62] S.M. Robinson. Generalized equations and their solutions. I. Basic theory. *Mathematical programming study*, 10:128–141, 1979.

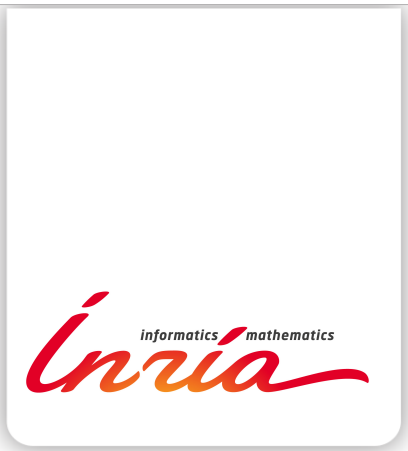
[63] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[64] M. Schatzman. A class of nonlinear differential equations of second order in time. *nla*, 2(3):355–373, 1978.

[65] D. Stewart. Reformulations of measure differential inclusions and their closed graph property. *jde*, 175(1):108–129, 2001.

[66] D. E. Stewart. Convergence of a time-stepping scheme for rigid body dynamics and resolution of Painlevé's problems. *Arch. Rat. Mech. Anal.*, 145(3):215–260, 1998.

[67] A. J. van der Schaft and J.M. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer Verlag London, 2000.



RESEARCH CENTRE
GRENOBLE – RHÔNE-ALPES

Inovallée
655 avenue de l'Europe Montbonnot
38334 Saint Ismier Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399